Mathematical Foundation for Computer Science

DEMTH403

Edited by:
Dr. Kulwinder Singh

LOVELY PROFESSIONAL UNIVERSITY

Mathematical Foundation for Computer Science

Edited By
Dr. Kulwinder Singh

Title: MATHEMATICAL FOUNDATION FOR COMPUTER SCIENCE

Author's Name: Dr. Nitin K. Mishra

Published By : Lovely Professional University

Publisher Address: Lovely Professional University, Jalandhar Delhi GT road, Phagwara - 144411

Printer Detail: Lovely Professional University

Edition Detail: (I)

ISBN: 978-93-94068-43-8

Copyrights@ Lovely Professional University

CONTENTS

Unit 1:	**The Foundations: Logic and Proofs – 1** *Dr. Nitin K. Mishra, Lovely Professional University*	1
Unit 2:	**Variables and Quantifiers** *Dr. Nitin K. Mishra, Lovely Professional University*	16
Unit 3:	**Partially Ordered Sets** *Dr. Nitin K. Mishra, Lovely Professional University*	33
Unit 4:	**Boolean Algebra** *Dr. Nitin K. Mishra, Lovely Professional University*	48
Unit 5:	**Introduction, Basic Counting Principles** *Dr. Nitin K. Mishra, Lovely Professional University*	57
Unit 6:	**Graphs Terminology** *Dr. Nitin K. Mishra, Lovely Professional University*	70
Unit 7:	**Graphs Terminology** *Dr. Nitin K. Mishra, Lovely Professional University*	89
Unit 8:	**Connectivity** *Dr. Nitin K. Mishra, Lovely Professional University*	113
Unit 9:	**Euler and Hamilton Paths** *Dr. Nitin K. Mishra, Lovely Professional University*	128
Unit 10:	**Shortest-Path Problems** *Dr. Nitin K. Mishra, Lovely Professional University*	146
Unit 11:	**Graph Coloring** *Dr. Nitin K. Mishra, Lovely Professional University*	173
Unit 12:	**Trees** *Dr. Nitin K. Mishra, Lovely Professional University*	186
Unit 13:	**Spanning Trees** *Dr. Nitin K. Mishra, Lovely Professional University*	199
Unit 14:	**Tree Traversal** *Dr. Nitin K. Mishra, Lovely Professional University*	225

Unit 1:The Foundations: Logic and Proofs - 1

CONTENTS

Objectives

Introduction

1.1 Propositional Logic

1.2 Conditional Statements

1.3 CONVERSE, CONTRAPOSITIVE, AND INVERSE

Summary

Answer for Self Assessment

Review Question's

Further Readings

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are: -

- Understand what is a proposition.
- Understand the negation of a proposition.
- Understand the conjunction of propositions.
- Understand the disjunction of propositions.
- Understand the use of negation with a conjunction as well as disjunction.
- Understand what is Conditional Statements.
- Understand the use of Conditional Statements in mathematical reasoning.
- Understand what is a biconditional Statements
- Learn what is the equivalence of formulas
- Understand what is duality law.

Introduction

A discrete mathematics course has more than one purpose. Students should learn a particular set of mathematical facts and how to apply them; more importantly, such a course should teach students how to think logically and mathematically. To achieve these goals, this text stresses mathematical reasoning and the different ways problems are solved. Five important themes are interwoven in this text: mathematical reasoning, combinatorial analysis, discrete structures, algorithmic thinking, and applications and modelling. A successful discrete mathematics course

should carefully blend and balance all five themes.

1.1 Propositional Logic

The rules of logic specify the meaning of mathematical statements. For instance, these rules help us understand and reason with statements such as "There exists an integer that is not the sum of two squares" and "For every positive integer n, the sum of the positive integers not exceeding n is $n(n + 1)/2$."

Logic is the basis of all mathematical reasoning and all automated reasoning. It has practical applications to the design of computing machines, to the specification of systems, to artificial intelligence, to computer programming, to programming languages, and to other areas of computer science, as well as to many other fields of study.

To understand mathematics, we must understand what makes up a correct mathematical argument, that is, proof. Once we prove a mathematical statement is true, we call it a theorem. A collection of theorems on a topic organize what we know about this topic. To learn a mathematical topic, a person needs to actively construct mathematical arguments on this topic, and not just read exposition. Moreover, knowing the proof of a theorem often makes it possible to modify the result to fit new situations.

Everyone knows that proofs are important throughout mathematics, but many people find it surprising how important proofs are in computer science. Proofs are used to verify that computer programs produce the correct output for all possible input values, to show that algorithms always produce the correct result, to establish the security of a system, and to create artificial intelligence. Furthermore, automated reasoning systems have been created to allow

computers to construct their own proofs.

In this section, we will explain what makes up a correct mathematical argument and introduce tools to construct these arguments. We will develop an arsenal of different proof methods that will enable us to prove many different types of results. After introducing many different methods of proof, we will introduce several strategies for constructing proofs. We will introduce the notion of a conjecture and explain the process of developing mathematics by studyingconjectures.

The rules of logic give a precise meaning to mathematical statements. These rules are used to distinguish between valid and invalid mathematical arguments. Because a major goal of this book is to teach the reader how to understand and how to construct correct mathematical arguments, we begin our study of discrete mathematics with an introduction to logic.

Besides the importance of logic in understanding mathematical reasoning, logic has numerous applications to computer science. These rules are used in the design of computer circuits, the construction of computer programs, the verification of the correctness of programs, and in many other ways. Furthermore, software systems have been developed for constructing some, but not all, types of proofs automatically. We will discuss these applications of logic in this and later chapters.

Propositions

Our discussion begins with an introduction to the basic building blocks of logic—propositions.

A proposition is a declarative sentence (that is, a sentence that declares a fact) that is either true

or false, but not both.

EXAMPLE 1 All the following declarative sentences are propositions.

1. Washington, D.C., is the capital of the United States of America.

2. Toronto is the capital of Canada.

3. 1 + 1 = 2.

4. 2 + 2 = 3.

Propositions 1 and 3 are true, whereas 2 and 4 are false.

▲

Some sentences that are not propositions are given in Example 2.

EXAMPLE 2 Consider the following sentences.

1. What time is it?

2. Read this carefully.

3. x + 1 = 2.

4. x + y = z.

Sentences 1 and 2 are not propositions because they are not declarative sentences. Sentences 3 and 4 are not propositions because they are neither true nor false. Note that each of sentences 3 and 4 can be turned into a proposition if we assign values to the variables. We will also discuss other ways to turn sentences such as these into propositions in Section 1.4. We use letters to denote propositional variables (or statement variables), that is, variables that represent propositions, just as letters are used to denote numerical variables. The conventional letters used for propositional variables are p, q, r, s, ... The truth value of a proposition is true, denoted by T, if it is a true proposition, and the truth value of a proposition is false, denoted by F if it is a false proposition.

The area of logic that deals with propositions are called propositional calculus or propositional logic. It was first developed systematically by the Greek philosopher Aristotle more than 2300 years ago.

We now turn our attention to methods for producing new propositions from those that we already have. These methods were discussed by the English mathematician George Boole in 1854 in his book The Laws of Thought. Many mathematical statements are constructed by

combining one or more propositions. New propositions, called compound propositions, are formed from existing propositions using logical operators.

DEFINITION 1 Let p be a proposition. The negation of p, denoted by¬p (also denoted by p), is the statement "It is not the case that p."

The proposition ¬p is read "not p." The truth value of the negation of p, ¬p, is the opposite of the truth value of p.

EXAMPLE 3 Find the negation of the proposition "Michael's PC runs Linux"

and express this in simple English.

Solution: The negation is "It is not the case that Michael's PC runs Linux." This negation can be more simply expressed as

"Michael's PC does not run Linux."

EXAMPLE 4 Find the negation of the proposition "Vandana's smartphone has at least 32GB of memory"

and express this in simple English.

Solution: The negation is "It is not the case that Vandana's smartphone has at least 32GB of memory." This negation can also be expressed as "Vandana's smartphone does not have at least 32GB of memory"

or even more simply as "Vandana's smartphone has less than 32GB of memory."

Table 1 displays the truth table for the negation of a proposition p. This table has a row for each of the two possible truth values of a proposition p. Each row shows the truth value of ¬p corresponding to the truth value of p for this row.

The negation of a proposition can also be considered the result of the operation of the negation operator on a proposition. The negation operator constructs a new proposition from a single existing proposition. We will now introduce the logical operators that are used to form

new propositions from two or more existing propositions. These logical operators are also called connectives.

Table 1

The Truth Table for the Negation of a Proposition.	
P	¬P
F	T
T	F

DEFINITION 2 Let p and q be propositions. The conjunction of p and q, denoted by p ∧ q, is the proposition "p and q." The conjunction p ∧ q is true when both p and q are true and is false otherwise. Table 2 displays the truth table of p ∧ q. This table has a row for each of the four possible combinations of truth values of p and q. The four rows correspond to the pairs of truth values

TT, TF, FT, and FF, where the first truth value in the pair is the truth value of p and the second truth value is the truth value of q.

Note that in logic the word "but" sometimes is used instead of "and" in conjunction. For example, the statement "The sun is shining, but it is raining" is another way of saying "The sun is shining and it is raining." (In natural language, there is a subtle difference in meaning between "and" and "but"; we will not be concerned with this nuance here.)

EXAMPLE 5 Find the conjunction of the propositions p and q where p is the proposition "Rebecca's PC has more than 16 GB free hard disk space" and q is the proposition "The processor in Rebecca's PC runs faster than 1 GHz."

Solution: The conjunction of these propositions, p ∧ q, is the proposition "Rebecca's PC has more than 16 GB free hard disk space, and the processor in Rebecca's PC runs faster than 1 GHz." This conjunction can be expressed more simply as "Rebecca's PC has more than 16 GB

free hard disk space, and its processor runs faster than 1 GHz." For this conjunction to be true, both conditions given must be true. It is false when one or both of these conditions are false. ▲

DEFINITION 3 Let p and q be propositions. The disjunction of p and q, denoted by p ∨ q, is the proposition "p or q." The disjunction p ∨ q is false when both p and q are false and is true otherwise. Table 3 displays the truth table for p ∨ q.

TABLE 2 The Truth Table for the Conjunction of Two Propositions.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

TABLE 3 The Truth Table for the Disjunction of Two Propositions.

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

EXAMPLE 6 What is the disjunction of the propositions p and q where p and q are the same propositions as in Example 5?

Solution: The disjunction of p and q, $p \vee q$, is the proposition "Rebecca's PC has at least 16 GB free hard disk space, or the processor in Rebecca's PC

runs faster than 1 GHz."

This proposition is true when Rebecca's PC has at least 16 GB free hard disk space, when the PC's processor runs faster than 1 GHz, and when both conditions are true. It is false when both of these conditions are false, that is, when Rebecca's PC has less than 16 GB free hard disk space and the processor in her PC runs at 1 GHz or slower.

1.2 Conditional Statements

We will discuss several other important ways in which propositions can be combined.

DEFINITION 5 Let p and q be propositions. The conditional statement p → q is the proposition "if p, then q." The conditional statement p → q is false when p is true and q is false, and true otherwise. In the conditional statement p → q, p is called the hypothesis (or antecedent or premise)

and q is called the conclusion (or consequence).

The statement p → q is called a conditional statement because p → q asserts that q is true on the condition that p holds. A conditional statement is also called an implication. The truth table for the conditional statement p → q is shown in Table 5. Note that the statement p → q is true when both p and q are true and when p is false (no matter what truth value q has).

TABLE 5 The Truth Table for the Conditional Statement $p \rightarrow q$.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Because conditional statements play such an essential role in mathematical reasoning, a variety of terminology is used to express p → q. You will encounter most if not all of the following ways to express this conditional statement: "if p, then q" "p implies q"

"if p, q" "p only if q"

"p is sufficient for q" "a sufficient condition for q is p"

"q if p" "q whenever p"

"q when p" "q is necessary for p"

"a necessary condition for p is q" "q follows from p"

"q unless ¬p"

A useful way to understand the truth value of a conditional statement is to think of an obligation or a contract. For example, the pledge many politicians make when running for office is "If I am elected, then I will lower taxes."

If the politician is elected, voters would expect this politician to lower taxes. Furthermore, if the politician is not elected, then voters will not have any expectation that this person will lower taxes, although the person may have sufficient influence to cause those in power to lower taxes.

It is only when the politician is elected but does not lower taxes that voters can say that the politician has broken the campaign pledge. This last scenario corresponds to the case when p is true but q is false in $p \rightarrow q$.

Similarly, consider a statement that a professor might make: "If you get 100% on the final, then you will get an A." If you manage to get 100% on the final, then you would expect to receive an A. If you do not get 100% you may or may not receive an A depending on other factors. However, if you do get 100%, but the professor does not give you an A, you will feel cheated. Of the various ways to express the conditional statement $p \rightarrow q$, the two that seem to cause the most confusion are "p only if q" and "q unless $\neg p$." Consequently, we will provide some guidance for clearing up this confusion.

To remember that "p only if q" expresses the same thing as "if p, then q," notes that "p only if q" says that p cannot be true when q is not true. That is, the statement is false if p is true, but q is false. When p is false, q may be either true or false, because the statement says nothing about the truth value of q. Be careful not to use "q only if p" to express p → q because this is incorrect. To see this, note that the true values of "q only if p" and p → q are different when p and q have different truth values.

To remember that "q unless ¬p" expresses the same conditional statement as "if p, then q," notes that "q unless ¬p" means that if ¬p is false, then q must be true. That is, the statement "q unless ¬p" is false when p is true but q is false, but it is true otherwise. Consequently, "q unless ¬p" and p → q always have the same truth value.

We illustrate the translation between conditional statements and English statements in Example 7.

EXAMPLE 7 Let p be the statement "Maria learns discrete mathematics" and q the statement "Maria will

find a good job." Express the statement p → q as a statement in English.

Solution: From the definition of conditional statements, we see that when p is the statement "Maria learns discrete mathematics" and q is the statement "Maria will find a good job," p → q represents the statement "If Maria learns discrete mathematics, then she will find a good job."

There are many other ways to express this conditional statement in English. Among the most natural of these are: "Maria will find a good job when she learns discrete mathematics."

"For Maria to get a good job, it is sufficient for her to learn discrete mathematics."

and

"Maria will find a good job unless she does not learn discrete mathematics."

▲

Note that the way we have defined conditional statements is more general than the meaning attached to such statements in the English language. For instance, the conditional statement in Example 7 and the statement "If it is sunny, then we will go to the beach."are statements used in a normal language where there is a relationship between the hypothesis and the conclusion. Further, the first of these statements is true unless Maria learns discrete mathematics, but she does not get a good job, and the second is true unless it is indeed sunny, but we do not go to the beach. On the other hand, the statement "If Juan has a smartphone, then 2 + 3 = 5" is true from the definition of a conditional statement, because its conclusion is true. (The truth value of the hypothesis does not matter then.) The conditional statement "If Juan has a smartphone, then 2 + 3 = 6"is true if Juan does not have a smartphone, even though 2 + 3 = 6 is false. We would not use these last two conditional statements in natural language (except perhaps in sarcasm), because there is no relationship between the hypothesis and the conclusion in either statement. In mathematical reasoning, we consider conditional statements of a more general sort than we use in English. The mathematical concept of a conditional statement is independent of a cause-and-effect relationship between hypothesis and conclusion. Our definition of a conditional statement specifies its truth values; it is not based on English usage. Propositional language is an artificial language; we only parallel English usage to make it easy to use and remember.

The if-then construction used in many programming languages is different from that used in logic. Most programming languages contain statements such as if p then S, where p is a proposition and S is a program segment (one or more statements to be executed).Whenthe execution of a program encounters such a statement, S is executed if p is true, but S is not executed if p is false, as illustrated in Example 8.

1.3 CONVERSE, CONTRAPOSITIVE, AND INVERSE

We can form some new conditional statements starting with a conditional statement $p \rightarrow q$. In particular, three related conditional statements occur so often that they have special names. The proposition $q \rightarrow p$ is called the converse of $p \rightarrow q$. The contrapositive of $p \rightarrow q$ is the proposition $\neg q \rightarrow \neg p$.

The proposition $\neg p \rightarrow \neg q$ is called the inverse of $p \rightarrow q$. We will see that of these three conditional statements formed from $p \rightarrow q$, only the contrapositive always has the same truth value as $p \rightarrow q$.

We first show that the contrapositive, $\neg q \rightarrow \neg p$, of a conditional statement $p \rightarrow q$ always has the same truth value as $p \rightarrow q$. To see this, note that the contrapositive is false only when $\neg p$ is false and $\neg q$ is true, that is, only when p is true and q is false.We now show that neither the converse, $q \rightarrow p$, nor the inverse, $\neg p \rightarrow \neg q$, has the same truth value as $p \rightarrow q$ for all possible truth values of p and q. Note that when p is true and q is false, the original conditional statement is false, but the converse and the inverse are both true.

When two compound propositions always have the same truth value we call them equivalent so that a conditional statement and its contrapositive are equivalent. The converse and the inverse of a conditional statement are also equivalent, as the reader can verify, but neither is equivalent to the original conditional statement. (We will study equivalent propositions in Section 1.3.) Take note that one of the most common logical errors is to assume that the converse or the inverse of a conditional statement is equivalent to this conditional statement.

We illustrate the use of conditional statements in Example 9.

EXAMPLE 9 What are the contrapositive, the converse, and the inverse of the conditional statement

"The home team wins whenever it is raining?"

Solution: Because "q whenever p" is one of the ways to express the conditional statement $p \rightarrow q$, the original statement can be rewritten as

"If it is raining, then the home team wins."

Consequently, the contrapositive of this conditional statement is "If the home team does not win, then it is not raining."

The converse is "If the home team wins, then it is raining." The inverse is"If it is not raining, then the home team does not win."

Only the contrapositive is equivalent to the original statement.

BICONDITIONALS

We now introduce another way to combine propositions that expresses that two propositions have the same truth value.

DEFINITION 6 Let p and q be propositions. The biconditional statement p ↔ q is the proposition "p if and only if q." The biconditional statement p ↔ q is true when p and q have the same truth values and is false otherwise. Biconditional statements are also called bi-implications.

The truth table for p ↔ q is shown in Table 6. Note that the statement p ↔ q is true when both the conditional statements p → q and q → p are true and is false otherwise. That is why we use the words "if and only if" to express this logical connective and why it is symbolically written

by combining the symbols→and←. There are some other common ways to express p ↔ q: "p is necessary and sufficient for q"

"if p then q, and conversely"

"p iff q."

The last way of expressing the biconditional statement p ↔ q uses the abbreviation "iff" for

"if and only if." Note that p ↔ q has the same truth value as (p → q) ∧ (q → p).

EXAMPLE 10 Let p be the statement "You can take the flight," and let q be the statement "You buy a ticket."

Then p ↔ q is the statement "You can take the flight if and only if you buy a ticket." This statement is true if p and q are either both true or both false, that is, if you buy a ticket and can take the flight or if you do not buy a ticket and you cannot take the flight. It is false when p and q have opposite truth values, that is, when you do not buy a ticket, but you can take the flight (such as when you get a free trip) and when you buy a ticket but you cannot take the flight (such as when the airline bumps you).

Truth Tables of Compound Propositions

We have now introduced four important logical connectives—conjunctions, disjunctions, conditional statements, and biconditional statements—as well as negations. We can use these connectives to build up complicated compound propositions involving any number of propositional variables. We can use truth tables to determine the truth values of these compound propositions, as Example 11 illustrates. We use a separate column to find the truth value of each compound expression that occurs in the compound proposition as it is built up. The truth values of the compound proposition for each combination of truth values of the propositional variables in it is found in the final column of the table.

EXAMPLE 11 Construct the truth table of the compound proposition

(p ∨¬q) → (p ∧ q).

Solution: Because this truth table involves two propositional variables p and q, there are four rows in this truth table, one for each of the pairs of truth values TT, TF, FT, and FF. The first two columns are used for the truth values of p and q, respectively. In the third column, we find the truth value of ¬q, needed to find the truth value of p ∨¬q, found in the fourth column. The fifth column gives the truth value of p ∧ q. Finally, the truth value of (p ∨¬q) → (p ∧ q) is found in the last column. The resulting truth table is shown in Table 7.

Table 7

p	q	$\neg q$	$p \vee \neg q$	$p \wedge q$	$(p \vee \neg q) \rightarrow (p \wedge q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

Propositional Equivalences

Introduction

An important type of step used in a mathematical argument is the replacement of a statement with another statement with the same truth value. Because of this, methods that produce propositions with the same truth value as a given compound proposition are used extensively in the construction of mathematical arguments. Note that we will use the term "compound proposition" to refer to an expression formed from propositional variables using logical operators, such as p ∧ q.

We begin our discussion with a classification of compound propositions according to their possible truth values.

DEFINITION 1 A compound proposition that is always true, no matter what the truth values of the propositional variables that occur in it, is called a tautology. A compound proposition that is always false is called a contradiction. A compound proposition that is neither a tautology nor a contradiction is called a contingency.

Tautologies and contradictions are often important in mathematical reasoning. Example 1 illustrates these types of compound propositions.

EXAMPLE 1 We can construct examples of tautologies and contradictions using just one propositional variable.

Consider the truth tables of p ∨¬p and p ∧¬p, shown in Table 1. Because p ∨¬p is always true, it is a tautology. Because p ∧¬p is always false, it is a contradiction.

Example of a Tautology

p	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

Example of a Contradiction.

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

Example of contingency

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Logical Equivalences

Compound propositions that have the same truth values in all possible cases are called logically equivalent. We can also define this notion as follows.

DEFINITION 2 The compound propositions p and q are called logically equivalent if p ↔ q is a tautology. The notation p ≡ q denotes that p and q are logically equivalent. Remark: The symbol ≡ is not a logical connective, and p ≡ q is not a compound proposition but rather is the statement that p ↔ q is a tautology. The symbol ⇔is sometimes used instead ≡ to denote logical equivalence.

One way to determine whether two compound propositions are equivalent is to use a truth table. In particular, the compound propositions p and q are equivalent if and only if the columns giving their truth values agree. Example 2 illustrates this method to establish an extremely important and useful logical equivalence, namely, that of ¬(p ∨ q) with ¬p ∧¬q.

EXAMPLE 2 Show that ¬(p ∨ q) and ¬p ∧¬q are logically equivalent.

Solution: The truth tables for these compound propositions are displayed in Table 3. Because the truth values of the compound propositions ¬(p ∨ q) and ¬p ∧¬q agree for all possible combinations of the truth values of P and q, it follows that¬(p ∨ q) ↔ (¬p ∧¬q) is a tautology and that these compound propositions are logically equivalent.

Duality

The identities in Table 5 come in pairs (except for the law of the double complement and the unit and zero properties). To explain the relationship between the two identities in each pair we use the concept of a dual. The dual of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0s and 1s.

The truth table for ¬ (p ∨ q) and ¬p ∧¬q is: -

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

EXAMPLE11

Find the duals of x (y + 0) and $x \cdot 1 + (y + z)$.

Solution: Interchanging · signs and + signs and interchanging 0s and 1s in these expressions produces their duals. The duals are x + (y · 1) and $(\bar{x} + 0)(\bar{y}\ z)$, respectively

Summary

The key concepts learned from this unit are: -

- We have learned what is a proposition.
- We have learned the negation of a proposition.
- We have learned the conjunction of propositions.
- We have learned the disjunction of propositions.
- We have learned the use of negation with a conjunction as well as disjunction.
- We have learned what is Conditional Statements.
- We have learned the use of Conditional Statements in mathematical reasoning.
- We have learned what is a biconditional Statements
- Learned what is the equivalence of formulas
- We have learned what is duality law.

Self Assessment

Q1. Which one of the following is a proposition?

a) How are you?
b) What time is it?
c) 4+x=5
d) India is in Europe.

Q2. What is the negation of the statement "Salman sent more than 100 text messages every day"?

a) Salman sent more than 200 text messages every day.
b) Salman sent less than 100 text messages but not every day.
c) Salman did not send more than 100 text messages every day.
d) Salman did not send any text message every day.

Q3. Select the appropriate option after evaluating the following four biconditionals are true or false.

1) 2 + 2 = 4 if and only if 1 + 1 = 2.

2) 1 + 1 = 2 if and only if 2 + 3 = 4.

3) 1 + 1 = 3 if and only if fishes can fly.

4) 0 > 1 if and only if 2 > 1.

a) Only 1 and 3 are True
b) Only option 3 and 4 are True
c) Option 1 is True
d) All options are false

Q4. What will be the Truth values of the statement $p \leftrightarrow \neg p$ for the Truth values T, F of p?

a) T, F
b) F, T
c) T, T
d) F, F

Q5. What will be the Truth values of the statement $(p \wedge q) \rightarrow (p \vee q)$ for the Truth values T, T, F, F of p and T, F, T, F of q?

a) T, F, T, F
b) F, T, F, T
c) T, T, T, T
d) F, F, F, F

Q6. If p: "You can use the wireless network in the airport," q: "You pay the daily fee," and r: "You are a subscriber to the service". Which is the right expression for the statement "To use the wireless network in the airport you must pay the daily fee unless you are a subscriber to the service".

a) $q \wedge r \rightarrow p$
b) $q \vee r \rightarrow p$
c) $p \wedge (q \vee r)$
d) $p \wedge (q \wedge r)$

Q7. What is the negation of the statement "Sam is rich and happy"?

a) Sam is poor and unhappy.
b) Either Sam is poor or happy
c) Either Sam is poor or unhappy
d) Sam is not rich and happy.

Q8. Let Q(x, y) denote the statement "y is the capital of x." What are these truth values? i) Q(Punjab, Chandigarh), ii) Q(India, New Delhi) iii) Q(Rajasthan, Shimla), iv) Q(Nepal, Kathmandu)

a) T,F,T,F
b) T,T,F,F
c) T,T,F,T
d) T,T,T,T

Q9. $(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$ is a

a) Contingency
b) Tautology
c) Contradiction
d) None of these

Q10. $(p \rightarrow q) \wedge (p \rightarrow r)$ is logically equivalent to

a) $p \rightarrow (q \vee r)$
b) $p \rightarrow (q \wedge r)$
c) $p \wedge (q \rightarrow r)$
d) $p \wedge \neg(q \rightarrow r)$

Q11. $\neg p \leftrightarrow q$ is logically equivalent to

a) $p \leftrightarrow \neg q$
b) $p \leftrightarrow q$
c) $p \wedge \neg q$
d) $p \vee \neg q$

Q12. $\neg p \leftrightarrow q$ is logically equivalent to

a) $p \leftrightarrow \neg q$
b) $p \leftrightarrow q$
c) $p \wedge \neg q$
d) $p \vee \neg q$

Q13. $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$ is a

a) Contingency
b) Contradiction
c) Tautology
d) All the above are true

Q14. If x and y are integers of opposite parity (one odd another even) the 5x+5y is

a) Always Odd
b) Always Even
c) Odd for some values and even for other values
d) Can not be decided

Q15. ¬(∀x ∈ A)p(x) is logically equivalent to

a) (∃ x ∈A)¬p(x)
b) (∃ x ∈¬A)p(x)
c) (∀ x ∈¬A)p(x)
d) (∀ x ∈A)¬p(x)

Q16. The contrapositive of the statement "If you are honest, then you are respected."

a) If You are honest then he is not respected.
b) If You are not respected than you are not honest.
c) If you are not honest then you are not respected.
d) If you are respected then you are honest.

Q17. The contrapositive of the statement "If Sahir is a poet, then he is poor"

a) If Sahir is rich then he is not a poet
b) If Sahir is not a poet then he is not poor
c) If Sahir is not poor then he is a poet
d) If Sahir is not a poet then he is not poor

Answer for Self Assessment

1.	D	2.	C	3.	B	4.	C	5.	D
6.	C	7.	B	8.	C	9.	C	10.	B
11.	B	12.	A	13.	B	14.	A	15.	A
16.	A	17.	B						

Review Question's

Q1. Which of these are propositions?What are the truth valuesof those that are propositions?

a) Do not pass go.

b) What time is it?

Q2. What is the negation of each of these propositions?

a) Jennifer and Teja are friends.

b) There are 13 items in a baker's dozen.

Q3. Suppose that SmartphoneAhas 256MBRAMand 32GBROM, and the resolution of its camera is 8 MP; SmartphoneB has 288 MB RAM and 64 GB ROM, and the resolution of its camera is 4 MP; and Smartphone C has128 MB RAM and 32 GB ROM, and the resolution ofits camera is 5 MP. Determine the truth value of each of these propositions.

a) SmartphoneBhas the mostRAMof these three smartphones.

b) Smartphone C has more ROM or a higher resolutioncamera than Smartphone B.

Q4. Let p and q be the propositions

p : I bought a lottery ticket this week.

q : I won the million dollar jackpot.

Express each of these propositions as an English sentence.

a) ¬p b) p ∨ q

Q6. Let p and q be the propositions "The election is decided"and "The votes have been counted," respectively. Expresseach of these compound propositions as an English sentence.

a) ¬p b) p ∨ q

Q7. Let p, q, and r be the propositions

p :You have the flu.

q :You miss the final examination.

r :You pass the course.

Express each of these propositions as an English sentence.

a) $p \rightarrow q$ b) $\neg q \leftrightarrow r$

Q8. Let p, q, and r be the propositions

p :You get an A on the final exam.

q :You do every exercise in this book.

r :You get an A in this class.

Further Readings

1. Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
2. Rosen, Kenneth H., and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.
3. Koshy, Thomas. *Discrete mathematics with applications*. Elsevier, 2004.
4. Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 02:Variables and Quantifiers

CONTENTS

Objectives

Introduction

2.1 Quantifiers

2.2 The universal quantifier

2.3 Some Terminology

2.4 Methods of Proving Theorems

2.5 Proof by Contraposition

2.6 Vacuous and Trivial Proofs

2.7 A Little Proof Strategy

2.8 Proofs by Contradiction

Summary

Self Assessment

Answer for Self Assessment

Review Questions

Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are: -

- understand what are Predicates.
- understand what is the statement function, variables
- understand binary, ternary and n – ary predicate.
- understand what truth value of the universal quantification.
- express the compound proposition with universal quantification in English.
- understand how to write compound proposition with universal quantification using disjunctions, conjunctions, and negations
- understand what truth value of the existential quantification.
- express the compound proposition with existential quantification in English.
- understand how to write compound proposition with existential quantification using disjunctions, conjunctions, and negations
- understand Methods of Proving Theorems.
- Understand what are Direct Proofs,
- Understand what is a Proof by Contraposition, Vacuous and Trivial proofs

Introduction

Propositional logic, studied in the previous unit, cannot adequately express the meaning of all statements in mathematics and natural language. For Example, suppose that we know that

"Every computer connected to the university network is functioning properly."

No rules of propositional logic allow us to conclude the truth of the statement

"MATH3 is functioning properly,"

where MATH3 is one of the computers connected to the university network. Likewise, we cannot use the rules of propositional logic to conclude from the statement

"CS2 is under attack by an intruder,"

where CS2 is a computer on the university network, to conclude the truth of

"There is a computer on the university network that is under attack by an intruder."

In this unit, we will introduce a more powerful type of logic called predicate logic. We will see how predicate logic can be used to express the meaning of a wide range of statements in mathematics and computer science in ways that permit us to reason and explore relationships between objects.

Subject Matter

To understand predicate logic, we first need to introduce the concept of a predicate. Afterwards, we will introduce the notion of quantifiers, which enable us to reason with statements that assert that a certain property holds for all objects of a certain type and with statements that assert the existence of an object with a particular property.

Predicates

Statements involving variables, such as

"x > 3," "x = y + 3," "x + y = z,"

and

"computer x is under attack by an intruder,"

and

"computer x is functioning properly,"

are often found in mathematical assertions, in computer programs, and system specifications. These statements are neither true nor false when the values of the variables are not specified. In this unit, we will discuss the ways that propositions can be produced from such statements. The statement "x is greater than 3" has two parts. The first part, the variable x, is the subject of the statement. The second part—the predicate, "is greater than 3"—refers to a property that the subject of the statement can have. We can denote the statement "x is greater than 3" by P (x), where P denotes the predicate "is greater than 3" and x is the variable. The statement P (x) is also said to be the value of the propositional function P at x. Once a value has been assigned to the variable x, the statement P (x) becomes a proposition and has a truth value. Consider Examples 1 and 2.

1. Let P (x) denote the statement "x > 3." What are the truth values of P (4) and P (2)?

Solution: We obtain the statement P (4) by setting x = 4 in the statement "x > 3." Hence, P (4), which is the statement "4 > 3," is true. However, P (2), which is the statement "2 > 3," is false.

Self-Assessment

1. Let P(x) denote the statement "$x \leq 4$." What are these truth values of P(0)?

2. Let P(x) denote the statement "$x \leq 4$." What are these truth values of P(4)?

3. Let P(x) denote the statement "$x \leq 4$." What are these truth values of P(6)?

2. Let A(x) denote the statement "Computer x is under attack by an intruder." Suppose that of the computers on campus, only CS2 and MATH1 are currently under attack by intruders. What are the truth values of A(CS1), A(CS2), and A(MATH1)?

Solution: We obtain statement A(CS1) by setting x = CS1 in the statement "Computer x is under attack by an intruder." Because CS1 is not on the list of computers currently under attack, we conclude that A(CS1) is false. Similarly, because CS2 and MATH1 are on the list of

computers under attack, we know that A(CS2) and A(MATH1) are true.

We can also have statements that involve more than one variable. For instance, consider the statement "x = y + 3." We can denote this statement by Q(x, y), where x and y are variables and Q is the predicate. When values are assigned to the variables x and y, the statement Q (x, y) has a truth value.

3. Let Q(x, y) denote the statement "x = y + 3." What are the truth values of the propositions Q(1, 2) and Q(3, 0)?

Solution: To obtain Q(1, 2), set x = 1 and y = 2 in the statement Q(x, y). Hence, Q(1, 2) is the statement "1 = 2 + 3," which is false. The statement Q(3, 0) is the proposition "3 = 0 + 3," which is true.

4. Let A(c, n) denote the statement "Computer c is connected to network n," where c is a variable representing a computer and n is a variable representing a network. Suppose that the computer MATH1 is connected to network CAMPUS2, but not to network CAMPUS1. What are the values of A(MATH1, CAMPUS1) and A(MATH1, CAMPUS2)?

Solution: Because MATH1 is not connected to the CAMPUS1 network, we see that A(MATH1, CAMPUS1) is false. However, because MATH1 is connected to the CAMPUS2 network, we see that A(MATH1, CAMPUS2) is true. Similarly, we can let R(x, y, z) denote the statement ``x + y = z." When values are assigned to the variables x, y, and z, this statement has a truth value.

5. What are the truth values of the propositions R(1, 2, 3) and R(0, 0, 1)?

Solution: The proposition R(1, 2, 3) is obtained by setting x = 1, y = 2, and z = 3 in the statement R(x, y, z). We see that R(1, 2, 3) is the statement "1 + 2 = 3," which is true. Also note that R(0, 0, 1), which is the statement "0 + 0 = 1," is false.

In general, a statement involving the n variables $x_1, x_2,..., x_n$ can be denoted by P ($x_1, x_2,..., x_n$). A statement of the form P ($x_1, x_2,..., x_n$) is the value of the propositional function P at the n-tuple ($x_1, x_2,..., x_n$), and P is also called an n place predicate or a nary predicate.

Self-Assessment

4. Let Q(x, y) denote the statement "x is the capital of y." What are these truth values of Q(Denver, Colorado)?

5. Let Q(x, y) denote the statement "x is the capital of y." What are these truth values of Q(Detroit, Michigan)

6. Let Q(x, y) denote the statement "x is the capital of y." What are these truth values of Q(Massachusetts, Boston)?

7. Let Q(x, y) denote the statement "x is the capital of y." What are these truth values of Q(NewYork, NewYork)?

2.1 Quantifiers

When the variables in a propositional function are assigned values, the resulting statement becomes a proposition with a certain truth value. However, there is another important way, called quantification, to create a proposition from a propositional function. Quantification expresses the extent to which a predicate is true over a range of elements. In English, the words all, some, many, none, and few are used in quantifications. We will focus on two types of quantification here: universal quantification, which tells us that a predicate is true for every element under consideration, and existential quantification, which tells us that there is one or more element under consideration for which the predicate is true. The area of logic that deals with predicates and quantifiers is called the predicate calculus.

2.2 The universal quantifier

Many mathematical statements assert that a property is true for all values of a variable in a particular domain, called the domain of discourse (or the universe of discourse), often just referred to as the domain. Such a statement is expressed

using universal quantification. The universal quantification of P (x) for a particular domain is the proposition that asserts that P (x) is true for all values of x in this domain. Note that the domain specifies the possible values of the variable x. The meaning of the universal quantification of P (x)

changes when we change the domain. The domain must always be specified when a universal quantifier is used; without it, the universal quantification of a statement is not defined.

Definition 1

The universal quantification of P (x) is the statement

"P (x) for all values of x in the domain."

The notation ∀x P(x) denotes the universal quantification of P (x). Here ∀ is called the universal quantifier. We read ∀x P(x) as "for all x P(x)" or "for every x P(x)." An element for which P (x) is false is called a counterExample of ∀x P(x).

The meaning of the universal quantifier is summarized in the first row of Table 1. We illustrate the use of the universal quantifier in Examples 8–13.

6. Let P (x) be the statement "x + 1 > x." What is the truth value of the quantification ∀x P(x), where the domain consists of all real numbers?

Solution: Because P (x) is true for all real numbers x, the quantification

∀x P(x)

is true.

Remark: Generally, an implicit assumption is made that all domains of discourse for quantifiers are nonempty. Note that if the domain is empty, then ∀x P(x) is true for any propositional function P (x) because there are no elements x in the domain for which P (x) is false

Besides "for all" and "for every," universal quantification can be expressed in many other ways, including "all of," "for each," "given any," "for arbitrary," "for each," and "for any."

Remark: It is best to avoid using "for any x" because it is often ambiguous as to whether "any" means "every" or "some." In some cases, "any" is unambiguous, such as when it is used in negatives, for example, "there is not any reason to avoid studying."

TABLE 1 Quantifiers.

Statement	When True?	When false?
∀x P(x).	P (x) is true for every x	There is an x for which P (x) is false.
∃x P(x)	There is an x for which P (x) is true.	P (x) is false for every x.

A statement ∀x P(x)is false, where P (x)is a propositional function, if and only if P (x)is not always true when x is in the domain. One way to show that P (x)is not always true when x is in the domain is to find a counterExample to the statement ∀x P(x). Note that a single counterExample is all we need to establish that ∀x P(x)is false. Example 7 illustrates how counterExamples are used.

7. Let Q(x) be the statement "x < 2." What is the truth value of the quantification ∀xQ(x), where the domain consists of all real numbers?

Solution: Q(x) is not true for every real number x, because, for instance, Q(3) is false. That is, x = 3 is a counterExample for the statement ∀xQ(x). Thus ∀xQ(x) is false.

8. Suppose that P (x) is "x2 > 0." To show that the statement ∀x P(x) is false where the universe of discourse consists of all integers, we give a counterExample. We see that x = 0 is a counterExample because x2 = 0 when x = 0, so that x2 is not greater than 0 when x = 0.

Looking for counterExamples to universally quantified statements is an important activity in the study of mathematics, as we will see in subsequent units of this book.

When all the elements in the domain can be listed—say, x_1, x_2,..., x_n —it follows that the universal quantification ∀x P(x) is the same as the conjunction

$P(x_1) \wedge P(x_2) \wedge \cdots \wedge P(x_n)$,

because this conjunction is true if and only if $P(x_1)$, $P(x_2)$, . . . , $P(x_n)$ are all true.

9. What is the truth value of ∀x P(x), where P (x) is the statement "$x^2 < 10$" and the domain consists of the positive integers not exceeding 4?

Solution: The statement ∀x P(x) is the same as the conjunction

P (1) ∧ P (2) ∧ P (3) ∧ P (4),

because the domain consists of the integers 1, 2, 3, and 4. Because P (4), which is the statement

"42 < 10," is false, it follows that ∀x P(x) is false.

10. What does the statement ∀xN (x) mean if N (x) is "Computer x is connected to the network" and the domain consists of all computers on campus?

Solution: The statement ∀xN (x) means that for every computer x on campus, that computer x is connected to the network. This statement can be expressed in English as "Every computer on campus is connected to the network."

As we have pointed out, specifying the domain is mandatory when quantifiers are used. The truth value of a quantified statement often depends on which elements are in this domain, as Example 11 shows.

11. What is the truth value of $\forall x(x^2 \geq x)$ if the domain consists of all real numbers? What is the truth value of this statement if the domain consists of all integers?

Solution: The universal quantification $\forall x(x^2 \geq x)$, where the domain consists of all real numbers, is false. For Example, $(1\ 2)^2 \geq 12$. Note that $x^2 \geq x$ if and only if $x^2 - x = x(x - 1) \geq 0$. Consequently, $x^2 \geq x$ if and only if $x \leq 0$ or $x \geq 1$. It follows that $\forall x(x2 \geq x)$ is false if the

domain consists of all real numbers (because the inequality is false for all real numbers x with $0 < x < 1$). However, if the domain consists of the integers, $\forall x(x2 \geq x)$ is true, because there are no integers x with $0 < x < 1$.

THE EXISTENTIAL QUANTIFIER Many mathematical statements assert that there is an element with a certain property. Such statements are expressed using existential quantification. With existential quantification, we form a proposition that is true if and only if P (x) is true for at least one value of x in the domain.

DEFINITION 2 The existential quantification of P (x) is the proposition

"There exists an element x in the domain such that P (x)."

We use the notation ∃x P(x) for the existential quantification of P (x). Here ∃ is called the existential quantifier. A domain must always be specified when a statement ∃x P(x) is used. Furthermore, the meaning of ∃x P(x) changes when the domain changes. Without specifying the domain, the statement ∃x P(x) has no meaning.

Besides the phrase "there exists," we can also express existential quantification in many other ways, such as by using the words "for some," "for at least one," or "there is." The existential quantification ∃x P(x) is read as

"There is an x such that P (x),"

"There is at least one x such that P (x),"

or

"For some x P(x)."

The meaning of the existential quantifier is summarized in the second row of Table 1. We illustrate the use of the existential quantifier in Examples 14–16.

12. Let P (x) denote the statement "x > 3." What is the truth value of the quantification ∃x P(x), where the domain consists of all real numbers?

Solution: Because "x > 3" is sometimes true—for instance, when x = 4—the existential quantification of P (x), which is ∃x P(x), is true.

Observe that the statement ∃x P(x) is false if and only if there is no element x in the domain for which P (x) is true. That is, ∃x P(x) is false if and only if P (x) is false for every element of the domain. We illustrate this observation in Example 13.

13. Let Q(x) denote the statement "x = x + 1." What is the truth value of the quantification ∃xQ(x), where the domain consists of all real numbers?

Solution: Because Q(x) is false for every real number x, the existential quantification of Q(x), which is ∃xQ(x), is false.

Remember that the truth value of ∃x P(x) depends on the domain!

Remark: Generally, an implicit assumption is made that all domains of discourse for quantifiers are nonempty. If the domain is empty, then ∃xQ(x) is false whenever Q(x) is a propositional function because when the domain is empty, there can be no element x in the domain for which Q(x) is true.

When all elements in the domain can be listed say, $x_1, x_2,..., x_n$ — the existential quantification ∃x P(x) is the same as the disjunction

$P(x_1) \vee P(x_2) \vee \cdots \vee P(x_n)$,

because this disjunction is true if and only if at least one of $P(x_1), P(x_2), \ldots, P(x_n)$ is true.

14. What is the truth value of ∃x P(x), where P (x) is the statement "$x^2 > 10$" and the universe of discourse consists of the positive integers not exceeding 4?

Solution: Because the domain is {1, 2, 3, 4}, the proposition ∃x P(x) is the same as the disjunction

P (1) ∨ P (2) ∨ P (3) ∨ P (4).

Because P (4), which is the statement "$4^2 > 10$," is true, it follows that ∃x P(x) is true.

It is sometimes helpful to think in terms of looping and searching when determining the truth value of a quantification. Suppose that there are n objects in the domain for the variable x. To determine whether ∀x P(x) is true, we can loop through all n values of x to see whether P (x) is always true. If we encounter a value x for which P (x) is false, then we have shown that

∀x P(x) is false. Otherwise, ∀x P(x) is true. To see whether ∃x P(x) is true, we loop through the n values of x searching for a value for which P (x) is true. If we find one, then ∃x P(x) is true. If we never find such an x, then we have determined that ∃x P(x) is false. (Note that this searching procedure does not apply if there are infinitely many values in the domain. However, it is still a useful way of thinking about the truth values of quantifications.)

Self-Assessment

8. Let P(x) be the statement "x spends more than five hours every weekday in class," where the domain for x consists of all students. Express ∃xP(x) quantifications in English.

9. Let P(x) be the statement "x spends more than five hours every weekday in class," where the domain for x consists of all students. Express ∀xP(x) quantifications in English.

10. Let P(x) be the statement "x spends more than five hours every weekday in class," where the domain for x consists of all students. Express ∃x ¬P(x) quantifications in English.

11. Let P(x) be the statement "x spends more than five hours every weekday in class," where the domain for x consists of all students. Express ∀x ¬P(x) quantifications in English.

12. Translate ∀x(C(x) → F(x)) statement into English, where C(x) is "x is a comedian" and F(x) is "x is funny" and the domain consists of all people.

13. Translate ∀x(C(x) ∧ F(x)) statement into English, where C(x) is "x is a comedian" and F(x) is "x is funny" and the domain consists of all people.

14. Translate ∃x(C(x) → F(x)) statements into English, where C(x) is "x is a comedian" and F(x) is "x is funny" and the domain consists of all people.

15. Translate ∃x(C(x) ∧ F(x)) statements into English, where C(x) is "x is a comedian" and F(x) is "x is funny" and the domain consists of all people.

Introduction to proofs.

Introduction

In this unit, we introduce the notion of proof and describe methods for constructing proofs. The proof is a valid argument that establishes the truth of a mathematical statement. A proof can use the hypotheses of the theorem, if any, axioms assumed to be true, and previously proven

theorems. Using these ingredients and rules of inference, the final step of the proof establishes the truth of the statement being proved.

In our discussion, we move from formal proofs of theorems toward more informal proofs. The arguments we introduced in Unit 1.6 to show that statements involving propositions and quantified statements are true were formal proofs, where all steps were supplied, and the rules for each step in the argument were given. However, formal proofs of useful theorems can

be extremely long and hard to follow. In practice, the proofs of theorems designed for human consumption are almost always informal proofs, where more than one rule of inference may

be used in each step, where steps may be skipped, where the axioms being assumed and the rules of inference used are not explicitly stated. Informal proofs can often explain to humans

why theorems are true, while computers are perfectly happy producing formal proofs using automated reasoning systems.

The methods of proof discussed in this chapter are important not only because they are used to prove mathematical theorems, but also for their many applications to computer science. These

applications include verifying that computer programs are correct, establishing that operating systems are secure, making inferences in artificial intelligence, showing that system specifica

tions are consistent, and so on. Consequently, understanding the techniques used in proofs is essential both in mathematics and computer science.

2.3 Some Terminology

Formally, a theorem is a statement that can be shown to be true. In mathematical writing, the term theorem is usually reserved for a statement that is considered at least somewhat important.

Less important theorems sometimes are called propositions. (Theorems can also be referred to as facts or results.) A theorem may be the universal quantification of a conditional statement with one or more premises and a conclusion. However, it may be some other type of logical statement, as the Examples later in this chapter will show. We demonstrate that a theorem is true

with proof. The proof is a valid argument that establishes the truth of a theorem. The statements used in a proof can include axioms (or postulates), which are statements we assume to be true

(forexample, the axioms for the real numbers, given in Appendix 1, and the axioms of plane geometry), the premises, if any, of the theorem, and previously proven theorems. Axioms may be stated using primitive terms that do not require definition, but all other terms used in theorems and their proofs must be defined. Rules of inference, together with definitions of terms, are used

to conclude from other assertions, tying together the steps of the proof. In practice, the final step of the proof is usually just the conclusion of the theorem. However, for clarity, we will

often recap the statement of the theorem as the final step of the proof. A less important theorem that is helpful in the proof of other results is called a lemma (plural lemmas or lemmata). Complicated proofs are usually easier to understand when they are proved using a series of lemmas, where each lemma is proved individually. A corollary is a theorem that can be established directly from a theorem that has been proved. A conjecture is a statement that is being proposed to be a true statement, usually based on some partial evidence, a heuristic argument, or the intuition of an expert. When proof of a conjecture is found, the conjecture becomes a theorem. Many times conjectures are shown to be false, so they are not theorems.

Before we introduce methods for proving theorems, we need to understand how many mathematical theorems are stated. Many theorems assert that a property holds for all elements in

a domain, such as the integers or the real numbers. Although the precise statement of such theorems needs to include a universal quantifier, the standard convention in mathematics is to

omit it.

For Example, the statement

"If x>y, where x and y are positive real numbers, then $x^2 > y^2$." really means

"For all positive real numbers x and y, if x>y, then $x^2 > y^2$." Furthermore, when theorems of this type are proved, the first step of the proof usually involves selecting a general element of the domain. Subsequent steps show that this element has the property in question. Finally, universal generalization implies that the theorem holds for all members of the domain.

2.4 Methods of Proving Theorems

Proving mathematical theorems can be difficult. To construct proofs we need all available ammunition, including a powerful battery of different proof methods. These methods provide the

overall approach and strategy of proofs. Understanding these methods is a key component of learning how to read and construct mathematical proofs. One we have chosen a proof method,

we use axioms, definitions of terms, previously proved results, and rules of inference to complete the proof. Note that in this book we will always assume the axioms for real numbers

found in Appendix 1. We will also assume the usual axioms whenever we prove a result about geometry. When you construct your proofs, be careful not to use anything but these axioms,

definitions, and previously proved results as facts! To prove a theorem of the form $\forall x(P(x) \rightarrow Q(x))$, our goal is to show that $P(c) \rightarrow Q(c)$ is true, where c is an arbitrary element of the domain, and then apply universal generalization.

In this proof, we need to show that a conditional statement is true. Because of this, we now focus on methods that show that conditional statements are true. Recall that $p \rightarrow q$ is true unless p is

true but q is false. Note that to prove the statement $p \rightarrow q$, we need only show that q is true if p is true. The following discussion will give the most common techniques for proving conditional

statements. Later we will discuss methods for proving other types of statements. In this unit, and in Unit 1.8, we will develop a large arsenal of proof techniques that can be used to prove

a wide variety of theorems.

When you read proofs, you will often find the words "obviously" or "clearly." These words indicate that steps have been omitted that the author expects the reader to be able to fill in.

Unfortunately, this assumption is often not warranted and readers are not at all sure how to fill in the gaps. We will assiduously try to avoid using these words and try not to omit too many steps.

However, if we included all steps in proofs, our proofs would often be excruciatingly long. Direct Proofs

A direct proof of a conditional statement $p \rightarrow q$ is constructed when the first step is the assumption that p is true; subsequent steps are constructed using rules of inference, with the final step showing that q must also be true. A direct proof shows that a conditional statement $p \rightarrow q$ is true by showing that if p is true, then q must also be true, so that the combination p true and q false never occurs. In direct proof, we assume that p is true and use axioms,

definitions, and previously proven theorems, together with rules of inference, to show that q must also be true. You will find that direct proofs of many results are quite straightforward, with a

fairly obvious sequence of steps leading from the hypothesis to the conclusion. However, direct proofs sometimes require particular insights and can be quite tricky. The first direct proofs we

present here are quite straightforward; later in the text, you will see some that are less obvious. We will provide examples of several different direct proofs. Before we give the first example,

we need to define some terminology.

Introduction to Proofs

DEFINITION 1 The integer n is even if there exists an integer k such that n = 2k and n is odd if there exists

an integer k such that n = 2k + 1. (Note that every integer is either even or odd, and no

integer is both even and odd.) Two integers have the same parity when both are even or both

are odd; they have opposite parity when one is even and the other is odd.

1. Give direct proof of the theorem "If n is an odd integer, then n2 is odd."

Solution: Note that this theorem states ∀nP ((n) → Q(n)), where P (n) is "n is an odd integer" and Q(n) is "n2 is odd." As we have said, we will follow the usual convention in mathematical

proofs by showing that P (n) implies Q(n), and not explicitly using universal instantiation. To begin a direct proof of this theorem, we assume that the hypothesis of this conditional statement

is true, namely, we assume that n is odd. By the definition of an odd integer, it follows that n = 2k + 1, where k is some integer. We want to show that n2 is also odd. We can square both sides of the equation n = 2k + 1 to obtain a new equation that expresses n2. When we do this, we find that $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$. By the definition of an odd integer, we can conclude that n^2 is an odd integer (it is more than twice an integer). Consequently, we have proved that if n is an odd integer, then n2 is an odd integer.

2. Give direct proof that if m and n are both perfect squares, then nm is also a perfect square.

(An integer a is a perfect square if there is an integer b such that a = b2.)

Solution: To produce direct proof of this theorem, we assume that the hypothesis of this conditional statement is true, namely, we assume that m and n are both perfect squares. By the

definition of a perfect square, it follows that there are integers s and t such that m = s2 and n = t2. The goal of the proof is to show that mn must also be a perfect square when m and n are;

looking ahead we see how we can show this by substituting s2 for m and t^2 for n into mn. This tells us that mn = s2t2. Hence, mn = s2t2 = (ss)(tt) = (st)(st) = (st)2, using commutativity

and associativity of multiplication. By the definition of a perfect square, it follows that mn is also a perfect square because it is the square of st, which is an integer. We have proved that if m

and n are both perfect squares, then mn is also a perfect square.

2.5 Proof by Contraposition

Direct proofs lead from the premises of a theorem to the conclusion. They begin with the premises, continue with a sequence of deductions, and end with the conclusion. However, we will see that attempts at direct proofs often reach dead ends. We need other methods of proving theorems of the form ∀x(P (x) → Q(x)). Proofs of theorems of this type that are not direct proofs, that is, that do not start with the premises and end with the conclusion, are called indirect proofs. An extremely useful type of indirect proof is known as proof by contraposition. Proofs by contraposition make use of the fact that the conditional statement p → ◌q is equivalent to its contrapositive, ¬q → ¬p. This means that the conditional statement p → q can be proved by showing that its contrapositive, ¬q → ¬p, is true. In a proof by contraposition of p → q, we take ¬q as a premise, and using axioms, definitions, and previously proven theorems, together with rules of inference, we show that ¬p must follow. We will illustrate proof by contraposition with two Examples. These Examples show that proof by contraposition can succeed when we cannot easily find direct proof.

3. Prove that if n is an integer and 3n + 2 is odd, then n is odd.

Solution: We first attempt a direct proof. To construct a direct proof, we first assume that 3n + 2 is an odd integer. This means that 3n + 2 = 2k + 1 for some integer k. Can we use this fact

to show that n is odd? We see that 3n + 1 = 2k, but there does not seem to be any direct way to conclude that n is odd. Because our attempt at a direct proof failed, we next try a proof by

contraposition.

The first step in a proof by contraposition is to assume that the conclusion of the conditional statement "If 3n + 2 is odd, then n is odd" is false; namely, assume that n is even. Then, by

the definition of an even integer, n = 2k for some integer k. Substituting 2k for n, we find that 3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1). This tells us that 3n + 2 is even (because it

is a multiple of 2), and therefore not odd. This is the negation of the premise of the theorem. Because the negation of the conclusion of the conditional statement implies that the hypothesis

is false, the original conditional statement is true. Our proof by contraposition succeeded; we have proved the theorem "If 3n + 2 is odd, then n is odd."

4. Prove that if n = ab, where a and b are positive integers, then a ≤ √n or b ≤ √n.

Solution: Because there is no obvious way of showing that a ≤ √n or b ≤ √n directly from the equation n = ab, where a and b are positive integers, we attempt a proof by contraposition. The first step in a proof by contraposition is to assume that the conclusion of the conditional statement "If n = ab, where a and b are positive integers, then a ≤ √n or b ≤ √n" is false. That is, we assume that the statement(a ≤ √n) ∨ (b ≤√n)is false. Using the meaning of disjunction together with De Morgan's law, we see that this implies that both a ≤ √n and b ≤ √n are false. This implies that a > √n and b > √n. We can multiply these inequalities together (using the fact that if 0 <s<t and 0 <u< v, then su< tv) to obtain ab > √n ·

√n = n. This shows that ab = n, which contradicts the statement n = ab.

Because the negation of the conclusion of the conditional statement implies that the hypothesis is false, the original conditional statement is true. Our proof by contraposition succeeded;

we have proved that if n = ab, where a and b are positive integers, then a ≤ √n or b ≤ √n.

2.6 Vacuous and Trivial Proofs

We can quickly prove that a conditional statement p → q is true when we know that p is false, because p → q must be true when p is false. Consequently, if we can show that p is false, then we have a proof, called a vacuous proof, of the conditional statement p → q. Vacuous proofs are often used to establish special cases of theorems that state that a conditional statement is true for all positive integers [i.e., a theorem of the kind ∀nP (n), where P (n) is a propositional function]. Proof techniques for theorems of this kind will be discussed in Unit 5.1.

5. Show that the proposition P (0) is true, where P (n) is "If n > 1, then n2 > n" and the domain consists of all integers.

Solution: Note that P (0) is "If 0 > 1, then 02 > 0." We can show P (0) using vacuous proof. Indeed, hypothesis 0 > 1 is false. This tells us that P (0) is automatically true.

Remark: The fact that the conclusion of this conditional statement, 02 > 0, is false is irrelevant to the truth value of the conditional statement because a conditional statement with a false hypothesis is guaranteed to be true. We can also quickly prove a conditional statement p → q if we know that the conclusion q is true. By showing that q is true, it follows that p → q must also be true. A proof of p → q that uses the fact that q is true is called trivial proof. Trivial proofs are often important when special cases of theorems are proved (see the discussion of proof by cases in Unit 1.8) and in mathematical induction, which is a proof technique discussed in Unit 5.1.

6. Let P (n) be "If a and b are positive integers with a ≥ b, then $a^n \geq b^n$," where the domain consists of all nonnegative integers. Show that P (0) is true.

Solution: The proposition P (0)is "If a ≥ b, then a0 ≥ b0." Because a0 = b0 = 1, the conclusion of the conditional statement "If a ≥ b, then a0 ≥ b0" is true. Hence, this conditional statement, which is P (0), is true. This is an example of trivial proof. Note that the hypothesis, which is the statement "a ≥ b," was not needed in this proof.

2.7 A Little Proof Strategy

We have described two important approaches for proving theorems of the form ∀x(P (x) → Q(x)): direct proof and proof by contraposition. We have also given examples that show how each is used. However, when you are presented with a theorem of the form ∀x(P (x) → Q(x)), which method should you use to attempt to prove it? We will provide a few rules of thumb here; in Unit 1.8 we will discuss proof strategy at greater length. When you want to prove a statement of the form ∀x(P (x) → Q(x)), first evaluate whether a direct proof looks promising. Begin by expanding the definitions in the hypotheses.

Start to reason using these hypotheses, together with axioms and available theorems. If a direct proof does not seem to go anywhere, try the same thing with proof by contraposition. Recall that in a proof by contraposition you assume that the conclusion of the conditional statement is false and use direct proof to show this implies that the hypothesis must be false. We illustrate

this strategy in Examples 7 and 8. Before we present our next Example, we need a definition.

DEFINITION 2 The real number r is rational if there exist integers p and q with q = 0 such that r = p/q. A real number that is not rational is called irrational.

7. Prove that the sum of two rational numbers is rational. (Note that if we include the implicit quantifiers here, the theorem we want to prove is "For every real number r and every real number s, if r and s are rational numbers, then r + s is rational.)

Solution: We first attempt a direct proof. To begin, suppose that r and s are rational numbers. From the definition of a rational number, it follows that there are integers p and q, with q

= 0, such that r = p/q, and integers t and u, with u = 0, such that s = t/u. Can we use this information to show that r + s is rational? The obvious next step is to add r = p/q and s = t/u, to obtain

r + s = p/q+t/u = pu + qt/qu .

Because q = 0 and u = 0, it follows that qu = 0. Consequently, we have expressed r + s as the ratio of two integers, pu + qt and qu, where qu = 0. This means that r + s is rational. We have proved that the sum of two rational numbers is rational; our attempt to find a direct proof succeeded.

8. Prove that if n is an integer and n2 is odd, then n is odd.

Solution: We first attempt a direct proof. Suppose that n is an integer and n2 is odd. Then, there exists an integer k such that n2 = 2k + 1. Can we use this information to show that n is odd?

There seems to be no obvious approach to show that n is odd because solving for n produces the equation $n = \pm\sqrt{2k + 1}$, which is not useful. Because this attempt to use direct proof did not bear fruit, we next attempt a proof by contraposition. We take as our hypothesis the statement that n is not odd. Because every integer is odd or even, this means that n is even. This implies that there exists an integer k such that n = 2k. To prove the theorem, we need to show that this hypothesis implies the conclusion that n2 is not odd, that is, that n2 is even. Can we use the equation n = 2k to achieve this? By squaring both sides of this equation, we obtain $n^2 = 4k^2 = 2(2k^2)$, which implies that n^2 is also even because $n^2 = 2t$, where $t = 2k^2$. We have proved that if n is an integer and n^2 is odd, then n is odd. Our attempt to find proof by contraposition succeeded.

2.8 Proofs by Contradiction

Suppose we want to prove that a statement p is true. Furthermore, suppose that we can find a contradiction q such that ¬p → q is true. Because q is false, but ¬p → q is true, we can conclude that ¬p is false, which means that p is true. How can we find a contradiction q that might help us prove that p is true in this way?

Because the statement r ∧¬r is a contradiction whenever r is a proposition, we can prove that p is true if we can show that ¬p → (r ∧¬r) is true for some proposition r. Proofs of this type are called proofs by contradiction. Because proof by contradiction does not prove a result directly, it is another type of indirect proof. We provide three examples of proof by contradiction.

The first is an example of an application of the pigeonhole principle, a combinatorial technique that we will cover in-depth in Unit 6.2.

9. Show that at least four of any 22 days must fall on the same day of the week.

Solution: Let p be the proposition "At least four of 22 chosen days fall on the same day of the week." Suppose that ¬p is true. This means that at most three of the 22 days fall on the same day of the week. Because there are seven days of the week, this implies that at most 21 days could have been chosen, as for each of the days of the week, at most three of the chosen days could fall on that day. This contradicts the premise that we have 22 days under consideration. That is, if r is the statement that 22 days are chosen, then we have shown that ¬p → (r ∧¬r). Consequently, we know that p is true. We have proved that at least four of 22 chosen days fall on the same day of the week.

10. Prove that √2 is irrational by giving proof by contradiction.

Solution: Let p be the proposition "√2 is irrational." To start a proof by contradiction, we suppose that ¬p is true. Note that ¬p is the statement "It is not the case that √2 is irrational," which says that √2 is rational. We will show that assuming that ¬p is true leads to a contradiction. If √2 is rational, there exist integers a and b with √2 = a/b, where b = 0 and a and b have no common factors (so that the fraction a/b is in lowest terms.) (Here, we are using the fact that every rational number can be written in lowest terms.) Because √2 = a/b, when both sides of this equation are squared, it follows that

2 = 2^a/b^2

Hence,

$2b^2 = a^2$.

By the definition of an even integer, it follows that a^2 is even. We next use the fact that if a^2 is even, a must also be even, which follows exercise 16. Furthermore, because a is even, by

the definition of an even integer, a = 2c for some integer c. Thus, $2b^2 = 4c^2$. Dividing both sides of this equation by 2 gives

$b^2 = 2c^2$.

By the definition of even, this means that b^2 is even. Again using the fact that if the square of an integer is even, then the integer itself must be even, we conclude that b must be even as well.

We have now shown that the assumption of ¬p leads to the equation √2 = a/b, where a and b have no common factors, but both a and b are even, that is, 2 divides both a and b. Note that the statement that √2 = a/b, where a and b have no common factors, means, in particular, that 2 does not divide both a and b. Because our assumption of ¬p leads to the contradiction that 2 divides both a and b and 2 does not divide both a and b, ¬p must be false. That is, the statement p, "√2 is irrational," is true. We have proved that √2 is irrational.

Proof by contradiction can be used to prove conditional statements. In such proofs, we first assume the negation of the conclusion. We then use the premises of the theorem and the negation of the conclusion to arrive at a contradiction. (The reason that such proofs are valid rests on the logical equivalence of p → q and (p ∧¬q) → F. To see that these statements are equivalent, simply note that

each is false in exactly one case, namely when p is true and q is false.) Note that we can rewrite a proof by contraposition of a conditional statement as proof by contradiction. In proof of $p \rightarrow q$ by contraposition, we assume that $\neg q$ is true. We then show that $\neg p$ must also be true. To rewrite a proof by contraposition of $p \rightarrow q$ as proof by contradiction, we suppose that both p and $\neg q$ are true. Then, we use the steps from the proof of $\neg q \rightarrow \neg p$ to show that $\neg p$ is true. This leads to the contradiction $p \wedge \neg p$, completing the proof. Example 9 illustrates how proof by contraposition of a conditional statement can be rewritten as proof by contradiction.

11. Give proof by contradiction of the theorem "If 3n + 2 is odd, then n is odd."

Solution: Let p be "3n + 2 is odd" and q be "n is odd." To construct proof by contradiction, assume that both p and $\neg q$ are true. That is, assume that 3n + 2 is odd and that n is not odd.

Because n is not odd, we know that it is even. Because n is even, there is an integer k such that n = 2k. This implies that 3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1). Because 3n + 2 is 2t, where t = 3k + 1, 3n + 2 is even. Note that the statement "3n + 2 is even" is equivalent to the statement $\neg p$, because an integer is even if and only if it is not odd. Because both p and $\neg p$ are true, we have a contradiction. This completes the proof by contradiction, proving that if 3n + 2 is odd, then n is odd.

Note that we can also prove by contradiction that $p \rightarrow q$ is true by assuming that p and $\neg q$ are true, and showing that q must also be true. This implies that $\neg q$ and q are both true, a contradiction. This observation tells us that we can turn direct proof into a proof by contradiction.

Summary

- The key concepts learned from this unit are: -
- We have learned what are Predicates.
- We have learned what is the statement function, variables
- We have learned binary, ternary and n – ary predicate.
- We have learned what truth value of the universal quantification and express the compound proposition with universal quantification in English.
- We have learned how to write compound proposition with universal quantification using disjunctions, conjunctions, and negations
- We have learned what truth value of existential quantification.
- express the compound proposition with existential quantification in English. We have learned how to write compound proposition with existential quantification using disjunctions, conjunctions, and negations.
- We have learned Methods of Proving Theorems.
- We have learned what are Direct Proofs.
- We have learned what is a Proof by Contraposition, Vacuous and Trivial proofs

Self Assessment

1. The $\neg p \leftrightarrow q$ is logically equivalent to
   A. $p \leftrightarrow \neg q$
   B. $p \leftrightarrow q$
   C. $p \wedge \neg q$
   D. $p \vee \neg q$

2. $\neg p \leftrightarrow q$ is logically equivalent to
   A. $p \leftrightarrow \neg q$
   B. $p \leftrightarrow q$

C. $p \wedge \neg q$

D. $p \vee \neg q$

3. $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$ is a
   A. Contingency
   B. Contradiction
   C. Tautology
   D. All the above are true

4. If x and y are integers of opposite parity (one odd another even) the 5x+5y is
   A. Always Odd
   B. Always Even
   C. Odd for some values and even for other values
   D. Can not be decided

5. $\neg(\forall x \in A)p(x)$ is logically equivalent to
   A. $(\exists x \in A)\neg p(x)$
   B. $(\exists x \in \neg A)p(x)$
   C. $(\forall x \in \neg A)p(x)$
   D. $(\forall x \in A)\neg p(x)$

6. Contrapositive of the statement "If you are honest, then you are respected."
   A. If You are honest then he is not respected.
   B. If You are not respected than you are not honest.
   C. If you are not honest then you are not respected.
   D. If you are respected then you are honest.

7. Contrapositive of the statement "If Sunil is a poet, then he is poor"
   A. If Sunil is rich then he is not poet
   B. If Sunil is not a poet then he is not poor
   C. If Sunil is not poor then he is a poet
   D. If Sunil is not a poet then he is not poor

8. In proving π as irrational, we begin with assumption $\sqrt{7}$ is rational in which type of proof?
   A. Direct proof
   B. Proof by Contradiction
   C. Vacuous proof
   D. Mathematical Induction

9. Which of the following can only be used in disproving the statements?
   A. Direct proof
   B. Contrapositive proofs
   C. Counter Example
   D. Mathematical Induction

10. Which one of the following is the most appropriate logical formula to represent the statement? "Students who know Mathematical , coding skills are placed"
    The following notations are used:

$M(x)$: *x is knowing the Mathematical skills*

$C(x)$: *x is knowing the Coding skills*

$P(x)$: *x is placed*

A. $\forall x(P(x) \rightarrow (M(x) \wedge C(x)))$
B. $\forall x((M(x) \wedge C(x)) \rightarrow P(x))$
C. $\exists x((M(x) \wedge C(x)) \rightarrow P(x)$
D. $\forall x((M(x) \vee C(x)) \rightarrow P(x))$

11. Consider the following assertion: "The two statements with *N is the set of natural numbers and Z is the set of integers*
    1. $\exists x \in D, (P(x) \wedge Q(x))$
    2. $(\exists x \in D, P(x)) \wedge (\exists x \in D, Q(x))$ *have the same truth value."*

    Which of the following is correct?

    A. *This assertion is false. A counterexample is D = N, P(x) = "x is divisible by 6," Q(x) = "x is divisible by 3."*
    B. *This assertion is true. The proof follows from the distributive law for ∧.*
    C. *This assertion is false. A counterexample is D = Z, P(x) = "x < 0," Q(x) = "x ≥ 0."*
    D. *This assertion is false. A counterexample is D = N, P(x) = "x is a square," Q(x) = "x is odd."*

12. In proving √10 as irrational, we begin with assumption √11 is rational in which type of proof?
    A. Direct proof
    B. Proof by Contradiction
    C. Vacuous proof
    D. Mathematical Induction

13. When to proof P→Q true, we proof P false, that type of proof is known as--------
    A. Direct proof
    B. Contrapositive proofs
    C. Vacuous proof
    D. Mathematical Induction

14. Let domain of m includes all students, P (m) be the statement "m spends more than 2 hours in playing polo". Express ∀m ¬P (m) quantification in English.
    A. A student is there who spends more than 2 hours in playing polo
    B. There is a student who does not spend more than 2 hours in playing polo
    C. All students spends more than 2 hours in playing polo
    D. No student spends more than 2 hours in playing polo

15. If R(x, y, z) denotes "$x^2 + y^2 = z^2$" then which of the following is correct?
    A. R(1, 2, 3) has truth value = true
    B. R(1, 0, 0) has truth value = true
    C. R(0, 1, 0) has truth value = true
    D. R(0, 0, 1) has truth value = false

Answer for Self Assessment

1.	A	2.	B	3.	A	4.	A	5.	A
6.	B	7.	A	8.	C	9.	C	10.	D
11.	C	12.	B	13.	C	14.	D	15.	D

Review Questions

1. T, since $0 \leq 4$

2. T, since $4 \leq 4$

3. F, since 6 is not ≤ 4

4. This is true.

5. This is false, since Lansing, not Detroit, is the capital.

6. This is false (but Q(Boston, Massachusetts) is true).

7. This is false, since Albany, not New York, is the capital.

8. There is a student who spends more than five hours every weekday in class.

9. Every student spends more than five hours every weekday in class.

10. There is a student who does not spend more than five hours every weekday in class.

11. No student spends more than five hours every weekday in class. (Or, equivalently, every student spends

less than or equal to five hours every weekday in class.)

12. This statement is that for every x, if x is a comedian, then x is funny. In English, this is most simply

stated, "Every comedian is funny."

13. This statement is that for every x in the domain (universe of discourse), x is a comedian and x is funny.

In English, this is most simply stated, "Every person is a funny comedian." Note that this is not the sort

of thing one wants to say. It makes no sense and doesn't say anything about the existence of boring

comedians; it's surely false because there exist lots of x for which C(x) is false. This illustrates the fact that

you rarely want to use conjunctions with universal quantifiers.

14. This statement is that there exists an x in the domain such that if x is a comedian then x is funny. In

English, this might be rendered, "There exists a person such that ifs/he is a comedian, then s/he is funny."

Note that this is not the sort of thing one wants to say. It makes no sense and doesn't say anything about

the existence of funny comedians; it's surely true because there exist lots of x for which C(x) is false (recall the definition of the truth value of p -+ q). This illustrates the fact that you rarely want to use conditional

statements with existential quantifiers.

15. This statement is that there exists an x in the domain such that x is a comedian and x is funny. In English, this might be rendered, ''There exists a funny comedian" or "Some comedians are funny" or "Some funny people are comedians."

Further Reading

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. *Discrete mathematics with applications*. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 03:Partially Ordered Sets

CONTENTS

Objectives
Introduction
3.1 Partial Orderings
3.2 Hasse Diagrams
3.3 Maximal and Minimal Elements
3.4 Lattices
Summary
Self Assessment
Answer for Self Assessment
Review Questions
Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are: -

- understand what are relations
- understand what are the different properties of relation
- understand what is reflexive, symmetric,
- understand what antisymmetric relation is.
- understand what transitive relation is.
- understand what are partial ordering
- understand what are comparable and incomparable elements
- understand what are total, well and lexicographic ordering
- understand how to represent relations using digraphs
- understand what is a Hasse Diagraph/Diagram
- understand how to find least upper bound and greatest lower bound
- understand what is a lattice
- understand what a bounded and distributive lattice is.

Introduction

3.1 Partial Orderings

We often use relations to order some or all of the elements of sets. For instance, we order words using the relation containing pairs of words (x, y), where x comes before y in the dictionary. We schedule projects using the relation consisting of pairs (x, y), where x and y are tasks I a project such that x must be completed before y begins. We order the set of integers using the relation containing the pairs (x, y), where x is less than y. When we add all of the pairs of the form (x, x) to these relations, we obtain a relation that is reflexive, antisymmetric, and transitive. These are properties that characterize relations used to order the elements of sets.

DEFINITION 1 A relation R on a set S is called a partial ordering or partial order if it is reflexive, antisymmetric, and transitive. A set S together with a partial ordering R is called a partially

orderedset, or poset, and is denoted by (S, R). Members of S are called elements of the poset. We give examples of posets in Examples 1–3

1. Show that the "greater than or equal" relation (≥) is a partial ordering on the set of integers.

Solution: Because a ≥ a for every integer a, ≥ is reflexive. If a ≥ b and b ≥ a, then a = b. Hence, ≥ is antisymmetric. Finally, ≥ is transitive because a ≥ b and b ≥ c simply that a ≥ c. It follows that ≥ is a partial ordering on the set of integers and (Z, ≥) is a poset. ▲

2. The divisibility relation | is a partial ordering on the set of positive integers, because it is reflexive, anti symmetric, and transitive. We see that (Z+, |) is a poset. Recall that (Z+ denotes the set of positive integers.) ▲

3. Show that the inclusion relation ⊆ is a partial ordering on the power set of a set S.

Solution: Because A ⊆ A whenever A is a subset of S, ⊆ is reflexive. It is antisymmetric because A ⊆ B and B ⊆ A imply that A = B. Finally, ⊆ is transitive, because A ⊆ B and B ⊆ C imply that A ⊆ C. Hence, ⊆ is a partial ordering on P (S), and (P (S), ⊆) is a poset. ▲

Example 4 illustrates a relation that is not a partial ordering.

4. Let R be the relation on the set of people such that xRy, if x and y are people and x, is older than y. Show that R is not a partial ordering.

Solution: Note that R is antisymmetric because if a person x is older than a person y, then y is not older than x. That is, if xRy, then yRx. The relation R is transitive because if person x is older than person y and y is older than person z, then x is older than z. That is, if xRy and yRz, then xRz. However, R is not reflexive, because no person is older than himself or herself. That is, xR x for all people x. It follows that R is not a partial ordering. ▲In different posets, different symbols such as ≤, ⊆, and |, are used for a partial ordering. However, we need a symbol that we can use when we discuss the ordering relation in an arbitrary poset. Customarily, the notation a ≤ b is used to denote that (a, b) ∈ R in an arbitrary poset (S, R). This notation is used because the "less than or equal to" relation on the set of real numbers is the most familiar example of a partial ordering and the symbol is similar to the symbol. (Note that the symbol is used to denote the relation in any poset, not just the "less than or equals" relation.) The notation a ≺ b denotes that a ≤ b, but a = b. Also, we say "a is less than b" or "b is greater than a" if a ≺ b. When a and b are elements of the poset (S, ≤), it is not necessary that either a ≤ b or b ≤ a. For instance, in (P (Z), ⊆), {1, 2} is not related to {1, 3}, and vice versa, because neither set is contained within the other. Similarly, in (Z+, |), 2 is not related to 3 and 3 is not related to 2, because 2 | 3 and 3 | 2. This leads to Definition 2.

Self-Assessment

Which of these relations on {0, 1, 2, and 3} are partial orderings? Determine the properties of a partial ordering that the others lack.

Q1. {(0,0),(1,1),(2,2),(3,3)}

Q2. {(0,0),(1,1),(2,0),(2,2),(2,3),(3,2),(3,3)}

Q3. {(0,0),(1,1),(1,2),(2,2),(3,3)}

Q4. {(0,0),(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)}

DEFINITION2 The elements a and b of a poset are called comparable if either a ≤ b or b ≤ a. When a and b are elements of S such that neither a ≤ b nor b ≤ a, a and b are called incomparable.

5. In the poset (Z+, |), are the integers 3 and 9 comparable? Are 5 and 7 comparable?

Solution: The integers 3 and 9 are comparable, because 3 | 9. The integers 5 and 7 are incomparable because 5 | 7 and 7 | 5. ▲

The adjective "partial" is used to describe partial orderings because pairs of elements may be incomparable. When every two elements in the set are comparable, the relation is called a total ordering.

Self-Assessment

Q5. Find two incomparable elements in the posets. (P ({0, 1, 2}), ⊆)

Q6. Find two incomparable elements in the posets. ({1, 2, 4, 6, and 8}, |)

DEFINITION 3 If (S, ≤) is a poset and every two elements of S are comparable, S is called a totally ordered or linearly ordered set, and is called a total order or a linear order. A totally ordered set is also called a chain.

6. The poset (Z, ≤) is totally ordered because a ≤ b or b ≤ a whenever a and b are integers. ▲

7. The poset (Z+, |) is not totally ordered because it contains elements that are incomparable, such as 5 and 7. ▲

In Chapter 6 we noted that (Z+, ≤) is well-ordered, where ≤ is the usual "less than or equal to" relation. We now define well-ordered sets.

DEFINITION 4 (S, ≤) is a well-ordered set if it is a poset such that is a total ordering and every nonempty subset of S has a least element.

8. The set of ordered pairs of positive integers, Z+× Z+, with (a_1, a_2) (b_1, b_2) if $a_1 < b_1$, or if $a_1 = b_1$ and $a_2 \leq b_2$ (the lexicographic ordering), is a well-ordered set. The set Z, with the usual ≤ ordering, is not well-ordered because the set of negative integers, which is a subset of Z, has no least element. We now state and prove that this proof technique is valid.

THEOREM 1 THE PRINCIPLE OF WELL-ORDERED INDUCTION Suppose that S is a well-

ordered set Then P (x) is true for all x ∈ S, if

INDUCTIVE STEP: For every y ∈ S, if P (x) is true for all x ∈ S with x ≺ y, then P (y) is true.

Proof: Suppose it is not the case that P (x) is true for all x ∈ S. Then there is an element y ∈ S such that P (y) is false. Consequently, the set A = {x ∈ S | P (x) is false} is nonempty.

Because S is well ordered, A has a least element a. By the choice of a as a least element of A, we know that P (x) is true for all x ∈ S with x ≺ a. This implies by the inductive step P (a) is true. This contradiction shows that P (x) must be true for all x ∈ S.

Remark: We do not need a basis step in a proof using the principle of well-ordered induction because if x0 is the least element of a well-ordered set, the inductive step tells us that P (x0) is true. This follows because there are no elements x ∈ S with $x \prec x_0$, so we know (using a vacuous proof) that P (x) is true for all x ∈ S with $x \prec x_0$. The principle of well-ordered induction is a versatile technique for proving results about well-ordered sets. Even when it is possible to use mathematical induction for the set of positive integers to prove a theorem, it may be simpler to use the principle of well-ordered induction, where we proved a result about the well-orderedset (N × N,) where ≺ is lexicographic ordering on N × N.

Lexicographic Order

The words in a dictionary are listed in alphabetic, or lexicographic, order, which is based on theordering of the letters in the alphabet. This is a special case of an ordering of strings on a set constructed from a partial ordering on the set. We will show how this construction works in any poset. First, we will show how to construct a partial ordering on the Cartesian product of two posets, (A1, 1) and (A2, 2). The lexicographic ordering on A1 × A2 is defined by specifying that one pair is less than a second pair if the first entry of the first pair is less than (in A1) the first entry of the second pair, or if the first entries are equal, but the second entry of this pair is less than (in A2)

the second entry of the second pair. In other words, (a1, a2) is less than (b1, b2), that is, (a1, a2) ≺ (b1, b2), either if a1 ≺1 b1 or if both a1 = b1 and a2 ≺2 b2. We obtain a partial ordering by adding equality to the ordering ≺ on A1 × A2.

9. Determine whether (3, 5) ≺ (4, 8), whether (3, 8) ≺ (4, 5), and whether (4, 9) ≺ (4, 11) in the poset (Z ×Z,≤), where is the lexicographic ordering constructed from the usual ≤ relation on Z.

Solution: Because 3 < 4, it follows that (3, 5) ≺ (4, 8) and that (3, 8) ≺ (4, 5). We have (4, 9) ≺ (4, 11), because the first entries of (4, 9) and (4, 11) are the same but 9 < 11. ▲

In Figure 1 the ordered pairs in $Z^+ \times Z^+$ that are less than (3, 4) are highlighted. A lexicographic ordering can be defined on the Cartesian product of n posets $(A_1, 1), (A_2, 2), \ldots, (A_n, n)$. Define the partial ordering on $A_1 \times A_2 \times \cdots \times A_n$ by $(a_1, a_2,\ldots,a_n) \prec (b_1, b_2,\ldots,b_n)$ if $a_1 \prec_1 b_1$, or if there is an integer $i > 0$ such that $a_1 = b_1,\ldots,a_i = b_i$, and $a_{i+1} \prec_{i+1} b_{i+1}$. In other words, one n-tuple is less than a second n-tuple if the entry of the first n-tuple in the first position where the two n-tuples disagree is less than the entry in that position in the second n-tuple.

(1, 7)(1, 6) (1, 5) (1, 4) (1, 3) (1, 2) (1, 1) (2, 7) (2, 6)

(2, 5) (2, 4)(2, 3) (2, 2) (2, 1) (3, 7)(3, 6) (3, 5) (3, 4)

(3, 3) (3, 2) (3, 1) (4, 7) (4, 6) (4, 5) (4, 4) (4, 3) (4, 2)

(4, 1)(5, 7)(5, 6)(5, 5) (5, 4) (5, 3)(5, 2)(5, 1)(6, 7)(6, 6)

(6, 5)(6, 4)(6, 3)(6, 2)(6, 1)(7, 7)(7, 6)(7, 5)(7, 4)(7, 3)

(7, 2)(7, 1). . .

. . .

. . .

. . .

. . .

. FIGURE 1 The Ordered Pairs Less Than (3, 4) in Lexicographic Order.

10. Note that (1, 2, 3, 5) ≺ (1, 2, 4, 3), because the entries in the first two positions of these 4-tuples agree, but in the third position the entry in the first 4-tuple, 3, is less than that in the second 4-tuple, 4. (Here the ordering on 4-tuples is the lexicographic ordering that comes from the usual "less than or equals" relation on the set of integers.) ▲

We can now define the lexicographic ordering of strings. Consider the strings $a_1a_2 \ldots a_m\ a_n\ b_1b_2 \ldots b_n$ on a partially ordered set S. suppose these strings are not equal. Let t be the minimum of m and n. The definition of lexicographic ordering is that the string $a_1a_2 \ldots a_m$ is less than $b_1b_2 \ldots b_n$ if and only if $(a_1, a_2,\ldots, a_t) \prec (b_1, b_2,\ldots, b_t)$, or $(a_1, a_2,\ldots, a_t) = (b_1, b_2,\ldots, b_t)$ and $m < n$, where ≺ in this inequality represents the lexicographic ordering of S^t. In other words, to determine the ordering of two different strings, the longer string is truncated to the length of the shorter string, namely, to t = min(m, n) terms. Then the t-tuples made up of the first t terms of each string are compared using the lexicographic ordering on S^t One string is less than another string if the t-tuple corresponding to the first string is less than the t-tuple of the second string, or if these two t-tuples are the same, but the second string is longer.

11. Consider the set of strings of lowercase English letters. Using the ordering of letters in the alphabet, a lexicographic ordering on the set of strings can be constructed. A string is less than a second-string if the letter in the first string in the first position where the strings differ comes before the letter in the second string in this position, or if the first string and the second string agree in all positions, but the second string has more letters. This ordering is the same as that used in dictionaries. For example, discreet ≺ discrete, because these strings differ first in the seventh position, and e ≺ t. Also, discreet ≺ discreetness, because the first eight letters agree, but the second string is longer. Furthermore, discrete ≺discretion, because discrete ≺ discretion

3.2 Hasse Diagrams

Many edges in the directed graph for a finite poset do not have to be shown because they must be present. For instance, consider the directed graph for the partial ordering $\{(a, b) \mid a \leq b\}$ on the set {1, 2, 3, 4}, shown in Figure 2(a). Because this relation is a partial ordering, it is reflexive, and its directed graph has loops at all vertices. Consequently, we do not have to show these loops because they must be present; in Figure 2(b) loops are not shown. Because a partial ordering is transitive, we do not have to show those edges that must be present because of transitivity. For example, in Figure 2(c) the edges (1, 3), (1, 4), and (2, 4) are not shown because they must be present. If we assume that all edges are pointed "upward" (as they are drawn in the figure), we do not have to show the directions of the edges; Figure 2(c) does not show directions. In general, we can represent a finite poset $(S, \leq)$ using this procedure: Start with the directed graph for this relation. Because a partial ordering is reflexive, a loop (a, a) is present at every vertex a. Remove these loops. Next, remove all edges that must be in partial ordering because of the presence of other edges and transitivity. That is, remove all edges (x, y) for which there is an element $z \in S$ such that $x \prec z$ and $z \prec x$. Finally, arrange each edge so that

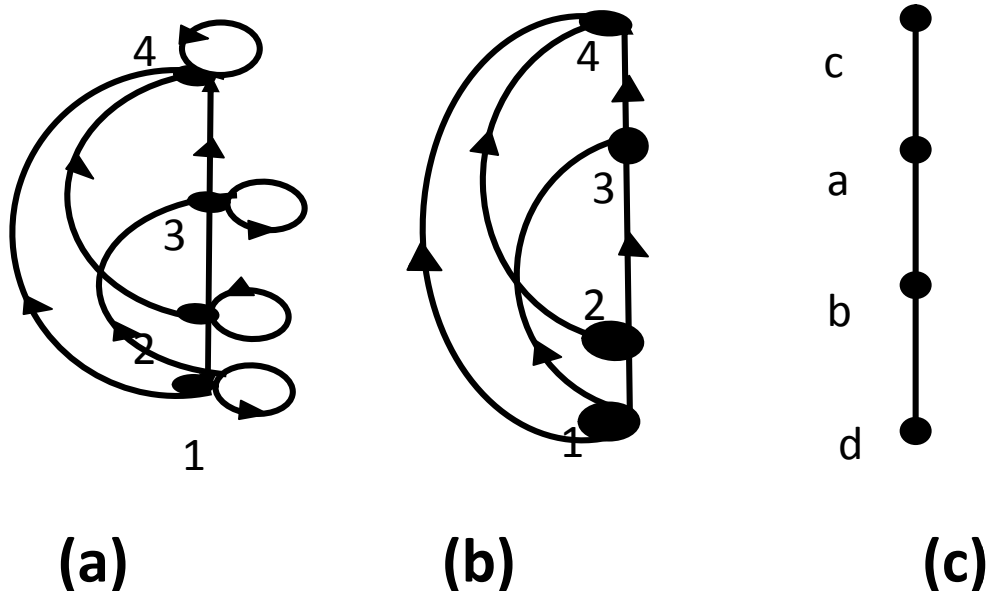


FIGURE2. Constructing the Hasse Diagramfor ({1, 2, 3, 4}, ≤).

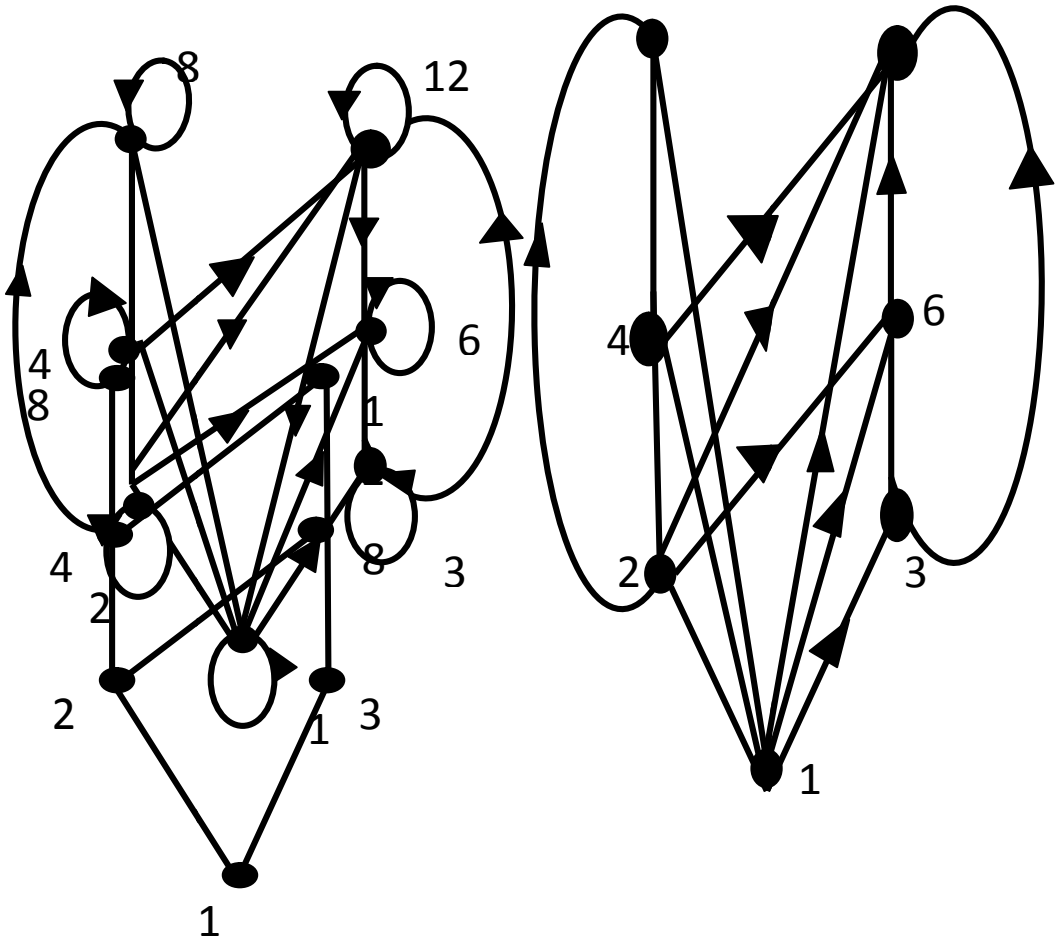
Its initial vertex is below its terminal vertex (as it is drawn on paper). Remove all the arrows on the directed edges, because all edges point "upward" toward their terminal vertex. These steps are well defined, and only a finite number of steps need to be carried out for a finite poset. When all the steps have been taken, the resulting diagram contains sufficient information to find the partial ordering, as we will explain later. The resulting diagram is called the Hasse diagram of $(S, \leq)$, named after the twentieth-century German mathematician Helmut Hasse who made extensive use of them.

Let $(S, \leq)$ be a poset. We say that an element $y \in S$ covers an element $x \in S$ if $x \prec y$ and there is no element $z \in S$ such that $x \prec z \prec y$. The set of pairs (x, y) such that y covers x is called the covering relation of $(S, \leq)$. From the description of the Hasse diagram of a poset, we see that the edges in the Hasse diagram of $(S, \leq)$ are upwardly pointing edges corresponding to the pairs in the covering relation of $(S, \leq)$. Furthermore, we can recover a poset from its covering relation, because it is the reflexive transitive closure of its covering relation. This tells us that we can construct a partial ordering from its Hasse diagram.

12. Draw the Hasse diagram representing the partial ordering {(a, b)|a divides b} on {1, 2, 3, 4, 6, 8, 12}.

Solution: Begin with the digraph for this partial order, as shown in Figure 3(a). Remove all loops, as shown in Figure 3(b). Then delete all the edges implied by the transitive property. These are (1, 4), (1, 6), (1, 8), (1, 12), (2, 8), (2, 12), and (3, 12). Arrange all edges to point upward, and delete all arrows to obtain the Hasse diagram. The resulting Hasse diagram is shown in Figure 3(c). ▲

13. Draw the Hasse diagram for the partial ordering {(A, B) | A ⊆ B} on the power set P (S) where S = {a, b, c}.



Solution: The Hasse diagram for this partial ordering is obtained from the associated digraph by deleting all the loops and all the edges that occur from transitivity, namely,(∅,{a, b}),(∅,{a, c}), (∅,{b, c}), (∅,{a, b, c}), ({a},{a, b, c}), ({b},{a, b, c}), and ({c},{a, b, c}). Finally, all edges point upward, and arrows are deleted. The resulting Hasse diagram is illustrated in Figure 4.

Self-Assessment

Q7. Draw the Hasse diagram for the "less than or equal to" relation on {0, 2, 5, 10, 11, 15}.

Q8. Draw the Hasse diagram for divisibility on the set {1, 2, 3, 4, 5, 6, 7, 8}

Q9. Draw the Hasse diagram for divisibility on the set {1, 2, 3, 5, 7, 11, 13}

Q10. Draw the Hasse diagram for divisibility on the set {1, 2, 3, 6, 12, 24, 36, 48}.

Q11. Draw the Hasse diagram for divisibility on the set {1, 2, 4, 8, 16, 32, 64}.

3.3 Maximal and Minimal Elements

Elements of posets that have certain extremely properties are important for many applications. An element of a poset is called maximal if it is not less than any element of the poset. That is, a is maximal in the poset (S, ≤) if there is no b ∈ S such that a ≺ b. Similarly, an element of a poset is called minimal if it is not greater than any element of the poset. That is, a is minimal if there is no element b ∈ S such that b ≺ a. Maximal and minimal elements are easy to spot using a Hasse diagram. They are the "top" and "bottom" elements in the diagram.

14. Which elements of the poset ({2, 4, 5, 10, 12, 20, 25}, |) are maximal, and which are minimal?

Solution: The Hasse diagram in Figure 5 for this poset shows that the maximal elements are 12, 20, and 25, and the minimal elements are 2 and 5. As this example shows, a poset can have more than one maximal element and more than one minimal element. ▲

Sometimes there is an element in a poset that is greater than every other element. Such an element is called the greatest element. That is, a is the greatest element of the poset (S, ≤)

Ifb ≤ a for all b ∈ S. The greatest element is unique when it exists.

Likewise, an element is called the least element if it is less than all the other elements in theposet. That is, a is the least element of (S, ≤) if a ≤ b for all b ∈ S. The least element is uniquewhen it exists.

FIGURE 4 the Hasse diagram of (p({a, b,c}),⊆).

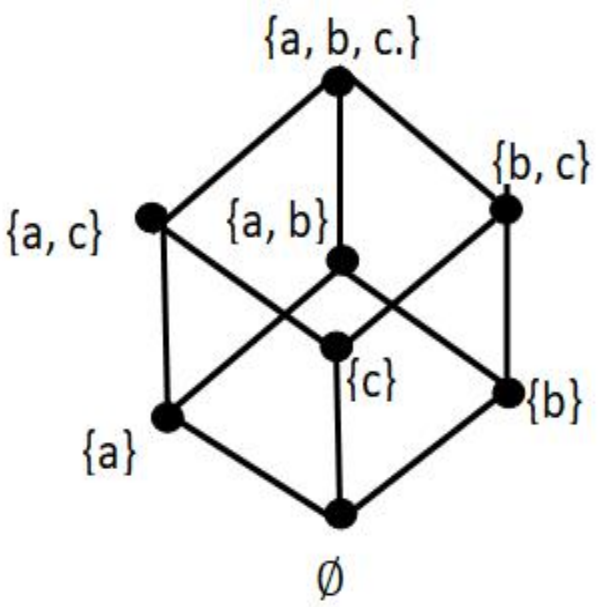


FIGURE 4 the Hasse diagram of (p({a,b,c}),⊆).

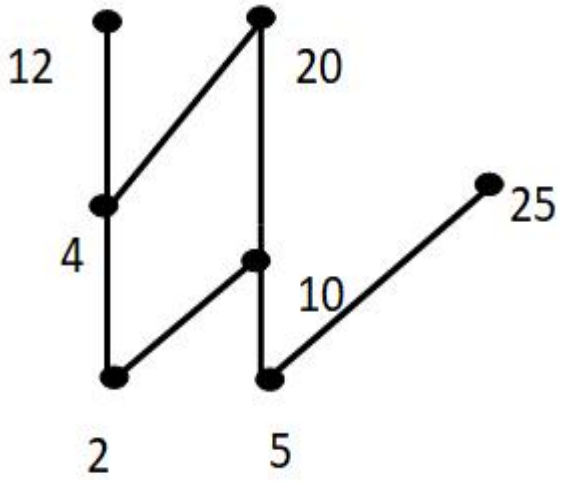


FIGURE 5 The Hasse diagram of a poset.

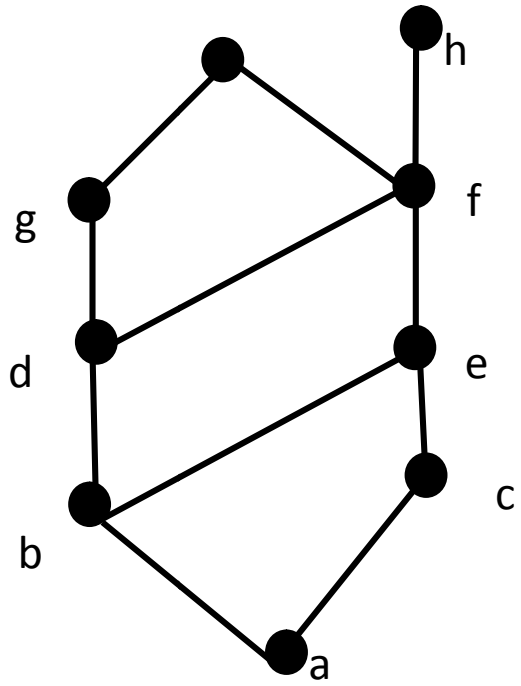


Figure 7 The Hasse Diagram of Poset.

15. Determine whether the posets represented by each of the Hasse diagrams in Figure 6 have the greatest element and a least element.

Solution: The least element of the poset with Hasse diagram (a) is a. This poset has no greatest element. The poset with Hasse diagram (b) has neither a least nor the greatest element. The poset with Hasse diagram (c) has no least element. Its greatest element is d. The poset with Hasse diagram (d) has least element a and greatest element d. ▲

16. Let S be a set. Determine whether there is the greatest element and a least element in the poset (P (S), ⊆).

Solution: The least element is the empty set, because ∅⊆ T for any subset T of S. The set S is the greatest element in this poset, because T ⊆ S whenever T is a subset of S. ▲

17. Is there a greatest element and a least element in the poset (Z+, |)?

Solution: The integer 1 is the least element because 1|n whenever n is a positive integer. Because there is no integer that is divisible by all positive integers, there is no greatest element. ▲

Sometimes it is possible to find an element that is greater than or equal to all the elements in a subset A of a poset (S, ≤). If u is an element of S such that a ≤ u for all elements a ∈ A, then u is called an upper bound of A. Likewise, there may be an element less than or equal to all the elements in A. If l is an element of S such that l ≤ a for all elements a ∈ A, then l is called a lower bound of A.

18. Find the lower and upper bounds of the subsets {a, b, c}, {j, h}, and {a, c, d, f } in the poset with the Hasse diagram shown in Figure 7.

Solution: The upper bounds of {a, b, c} are e, f, j, and h, and it's only lower bound is a. There are no upper bounds of {j, h}, and its lower bounds are a, b, c, d, e, and f. The upper bounds of {a, c, d, f} are f , h, and j , and its lower bound is a.

The element x is called the least upper bound of the subset A if x is an upper bound that is less than every other upper bound of A. Because there is only one such element, if it exists, it makes sense to call this element the least upper bound. That is, x is the least upper bound of A if a ≤ x whenever a

∈ A, and x ≤ z whenever z is an upper bound of A. Similarly, the element y is called the greatest lower bound of A if y is a lower bound ofA and z ≤ y whenever z is a lower bound of A. The greatest lower bound of A is unique if it exists. The greatest lower bound and least upper bound of a subset A are denoted by glb(A) and lub(A), respectively.

19. Find the greatest lower bound and the least upper bound of {b, d, g}, if they exist, in the poset shown in Figure 7.

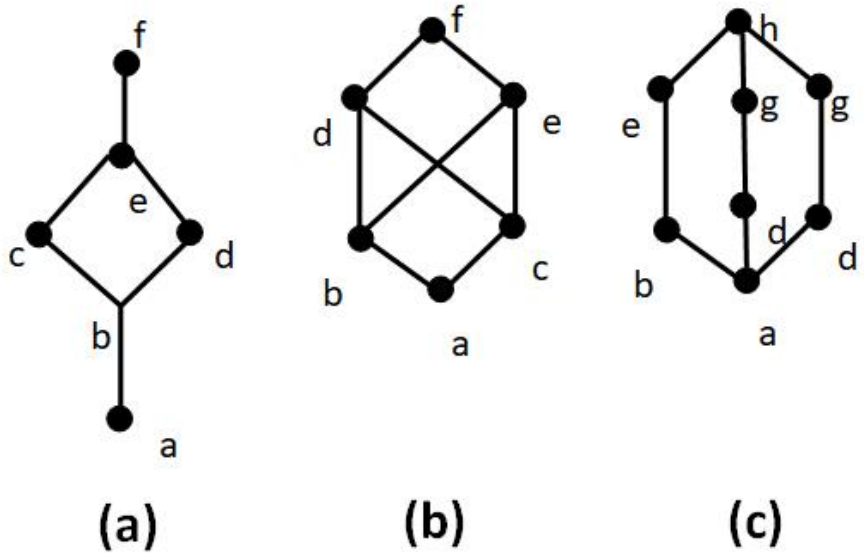
Solution: The upper bounds of {b, d, g} are g and h. Because g ≺ h, g is the least upper bound.

The lower bounds of {b, d, g} are a and b. Because a ≺ b, b is the greatest lower bound. ▲

20. Find the greatest lower bound and the least upper bound of the sets{3, 9, 12} and {1, 2, 4, 5, 10},if they exist, in the poset (Z+, |).

Solution: An integer is a lower bound of {3, 9, 12} if 3, 9, and 12 are divisible by this integer. The only such integers are 1 and 3. Because 1 | 3, 3 is the greatest lower bound of {3, 9, 12}. The only lower bound for the set {1, 2, 4, 5, 10} with respect to | is the element 1. Hence, 1 is the greatest lower bound for {1, 2, 4, 5, 10}. An integer is an upper bound for {3, 9, 12} if and only if it is divisible by 3, 9, and 12. The integers with this property are those divisible by the least common multiple of 3, 9, and 12, which is 36. Hence, 36 is the least upper bound of {3, 9, 12}. A positive integer is an upper bound for the set {1, 2, 4, 5, 10} if and only if it is divisible by 1, 2, 4, 5, and 10. The integers with this property are those integers divisible by the least common multiple of these integers,which is 20. Hence, 20 is the least upper bound of {1, 2, 4, 5, 10}.

Figure 8 Hasse diagram of three poset



Self-Assessment

Q12. Find the maximal elements for the poset ({3, 5,9,15, 24, 45}

Q13. Find the minimal elements for the poset ({3, 5,9,15, 24, 45},|)

Q14. Find the greatest element for the poset ({3, 5,9,15, 24, 45},|)

Q15. Find the least element for the poset ({3, 5,9,15, 24, 45},|)

3.4 Lattices

A partially ordered set in which every pair of elements has both a least upper bound and a greatest lower bound is called a lattice. Lattices have many special properties. Furthermore, lattices are used in many different applications such as models of information flow and play an important role in Boolean algebra.

21. Determine whether the posets represented by each of the Hasse diagrams in Figure 8 are lattices.

Solution: The posets represented by the Hasse diagrams in (a) and (c) are both lattices because in each poset every pair of elements has both a least upper bound and a greatest lower bound, as the reader should verify. On the other hand, the poset with the Hasse diagram shown in (b) is not a lattice, because the elements b and c have no least upper bound. To see this, note that each of the elements d, e, and f is an upper bound, but none of these three elements precedes the other two with respect to the ordering of this poset. ▲

22. Is the poset (Z+, |) a lattice?

Solution: Let a and b be two positive integers. The least upper bound and greatest lower bound of these two integers are the least common multiple and the greatest common divisor of these integers, respectively, as the reader should verify. It follows that this poset is a lattice. ▲

23. Determine whether the posets ({1, 2, 3, 4, 5}, |) and ({1, 2, 4, 8, 16}, |) are lattices.

Solution: Because 2 and 3 have no upper bounds in ({1, 2, 3, 4, 5}, |), they certainly do not have the least upper bound. Hence, the first poset is not a lattice. Every two elements of the second poset have both a least upper bound and a greatest lower bound. The least upper bound of two elements in this poset is the larger of the elements and the greatest lower bound of two elements is the smaller of the elements, as the reader should verify. Hence, this second poset is a lattice.

24. Determine whether (P (S), ⊆) is a lattice where S is a set.

Solution: Let A and B be two subsets of S. The least upper bound and the greatest lower bound of A and B are A ∪ B and A ∩ B, respectively, as the reader can show. Hence, (P (S), ⊆) is a lattice.

Let L be a lattice. Define the meet (∧) and join (∨) operations by x ∧ y = glb(x, y) and x ∨ y = lub(x, y).

A lattice L is bounded if it has both an upper bound, de- noted by 1, such that x ≤ 1 for all x ∈ L and a lower bound,denoted by 0, such that 0 ≤ x for all x ∈ L.

A lattice is called distributive if x ∨ (y ∧ z) = (x ∨ y) ∧ (x ∨ z) and x ∧ (y ∨ z) = (x ∧ y) ∨ (x ∧ z) for all x, y, and z in L.

The complement of an element a of a bounded lattice L with upper bound 1 and lower bound 0 is an element b such that a ∨ b = 1 and a ∧ b = 0. Such a lattice is complemented if every element of the lattice has a complement.

Summary

The key concepts learned from this unit are: -

- We have learned what are relations
- We have learned what are the different properties of relation
- We have learned what is reflexive, symmetric,
- We have learned what antisymmetric relation is.
- We have learned what transitive relation is.

- We have learned what are partial ordering
- We have learned what are comparable and incomparable elements
- We have learned whatare total, well and lexicographic ordering
- We have learned how to represent relations using digraphs
- We have learned what is a Hasse Diagraph/Diagram
- We have learned how to find least upper bound and greatest lower bound
- We have learned what is a lattice
- We have learned what a bounded and distributive lattice is.

Self Assessment

1. The ordered pairs {(0,0), (1, 1), (2,2), (3,3)}, in the relation R from A = {0, 1, 2, 3, 4} to B = {0, 1, 2, 3}, where (a, b) ∈ R are if and only if

A. a = b
B. a + b = 4.
C. a > b
D. a | b.

2. Which of the relations on the set {1, 2, 3, 4}, is antisymmetric?

A. {*(1, 2), (2, 3), (3, 4)*}
B. {*(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)*}
C. {*(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)*}
D. {*(2, 4), (4, 2)*}

3. The ordered pairs {(0, 1), (1, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (4, 1), (4, 3)}, in the relation *R* from *A* = {0, 1, 2, 3, 4} to *B* = {0, 1, 2, 3}, where *(a, b)* ∈*R are* if and only if

A. *a > b*.
B. *a* | *b*.
C. gcd*(a, b)* = 1
D. lcm*(a, b)* = 2.

4. Which of the relations on the set {1, 2, 3, 4}, is not transitive?

A. {(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)}
B. {(1, 1), (2, 2), (3, 3), (4, 4)}
C. {(2, 4), (4, 2)}
D. {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}

5. Which of the relations on the set {1, 2, 3, 4}, is not reflexive?

A. {(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (3, 4)}
B. {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}
C. {(1, 1), (2, 2), (3, 3), (4, 4)}
D. {(1, 1), (1, 3), (1, 4), (1, 2), (2, 3), (2, 4), (2, 1), (2, 2), (3, 3), (3, 1), (3, 4), (4, 4)}

6. The two incomparable elements in the POSET (P({1,2,3}),⊆) are
   A. {1} and {2}
   B. b) {2} and {2,3}
   C. c) {1} and {3,1}
   D. d) All of the above

7. The maximal elements for the POSET ({{5}, {6}, {8}, {5, 6}, {5, 8}, {6, 8}, {7, 8}, {5, 7, 8}, {6, 7, 8}}, ⊆) are
   A. {6}, {5}, and {7,8}.
   B. {5}, {5,7,8}, and {6,7,8}.
   C. {5,6}, {5,7,8}, and {6,7,8}.
   D. {5,6}, {5,7,8}, and {8,8,8}.

8. The minimal elements for the POSET ({{0}, {2}, {6},{0, 2}, {0, 6}, {2, 6}, {4, 6}, {0, 4, 6}, {2, 4, 6}}, ⊆) are
   A. {2}, {6}, and {4}.
   B. {0}, {4}, and {2,4,6}.
   C. { 0} , { 2} , and { 6}
   D. {0,2}, {0,4,6}, and {2,4,6}.

9. The greatest lower bound of {{5, 7, 8}, {6, 7, 8}}, elements for the POSET ({{5}, {6}, {8},{5, 6}, {5, 8}, {6, 8}, {7, 8}, {5, 7, 8}, {6, 7, 8}}, ⊆) are
   A. {7,8}.
   B. {5}
   C. {6,7,8}
   D. {5,7,8}

10. The least upper bound of {{2}, {6}}, elements for the POSET ({{0}, {2}, {6}, {0, 2}, {0, 6}, {2, 6}, {4, 6}, {0, 4, 6}, {2, 4, 6}}, ⊆) are
    A. {2}
    B. {2, 4, 6}.
    C. { 0}
    D. {2, 6}.

11. Z,≠) is not a POSET as the relation is not
    A. Reflexive
    B. Antisymmetric
    C. Transitive
    D. All of the above

12. Which of the relations on the set {1, 2, 3, 4}, is neither symmetric nor antisymmetric?
    A. xy ≥ 1.
    B. x = y2.
    C. x is a multiple of y.
    D. x and y are both negative or both nonnegative.

13. Which of these relations on {0, 1, 3, 4} are partial orderings?
    A. {(0, 0), (1, 1), (3, 0), (3, 3), (3, 4), (4, 3), (4, 4)}
    B. {(0, 0), (1, 1), (1, 3), (3, 3), (4, 4)}
    C. {(0, 0), (0, 1), (0, 3), (1, 0), (1, 1), (1, 3), (3, 0), (3, 3), (4, 4)}
    D. None of the above

14. Which of the relations on the set {1, 2, 3, 4}, is not antisymmetric?
    A. {(1, 2), (2, 3), (3, 4)}
    B. {(1, 1), (2, 2), (3, 3), (4, 4)}
    C. {(1, 3), (1, 4), (2, 3), (2, 4), (3, 1), (3, 4)}
    D. {(1, 1), (2, 2), (3, 3), (4, 4),(5,5)}

15. Which of the relations on the set {1, 2, 3, 4}, is neither symmetric nor antisymmetric?
    A. {(2, 4), (4, 2)}
    B. {(1, 1), (2, 2), (3, 3), (4, 4)}
    C. {(1, 3), (1, 4), (2, 3), (2, 4), (3, 1), (3, 4)}
    D. {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}

Answer for Self Assessment

1.	A	2.	A	3.	C	4.	C	5.	A
6.	A	7.	C	8.	C	9.	A	10.	D
11.	A	12.	C	13.	B	14.	C	15.	C

Review Questions

1. Clearly this relation is reflexive because each of 0, 1, 2, and 3 is related to itself. The relation is also antisymmetric because the only way for a to be related to b is for a to equal b. Similarly, the relation is transitive, because if a is related to b, and b is related to c, then necessarily a= b = c so a is related to c (because the relation is reflexive). This is just the equality relation on {O, 1, 2, 3}; more generally, the equality relation on any set satisfies all three conditions and is, therefore, a partial ordering. (It is the smallest partial ordering; reflexivity insures that every partial ordering contains at least all the pairs (a, a) .)

2. This is not a partial ordering, because although the relation is reflexive, it is not antisymmetric (we have 2 R 3 and 3 R 2, but $2 \neq 3$), and not transitive (3 R 2 and 2 R 0, but 3 is not related to 0).

3. This is a partial ordering because it is clearly reflexive; is antisymmetric (we just need to note that (1, 2) is the only pair in the relation with unequal components); and is transitive (for the same reason).

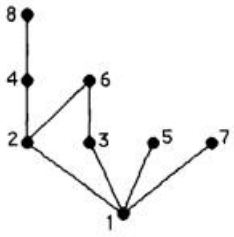
4. This is a partial ordering because it is the "less than or equal to" relation on {1, 2, 3} together with the isolated point 0. e) This is not a partial ordering. The relation is clearly reflexive, but it is not antisymmetric (0 R 1 and 1 R 0, but $0 \neq 1$) and not transitive (2 R 0 and 0R1, but 2 is not related to 1).

5. We need to find elements such that the relation holds in neither direction between them. The answers we give are not the only ones possible. One such pair is {1} and {2}. These are both subsets of {0, 1, 2}, so they are in the poset, but neither is a subset of the other.

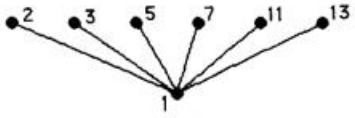
6. Neither 6 nor 8 divides the other, so they are incomparable.

7. This is a totally ordered set, so the Hasse diagram is linear.

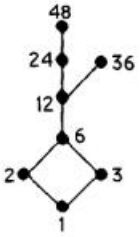
8.



9.



10.



11.

12. Maximal elements are those that do not divide any other elements of the set. In this case, 24 and 45 are the only numbers that meet that requirement.

13. Minimal elements are those that are not divisible by any other elements of the set. In this case, 3 and 5 are the only numbers that meet that requirement.

14. A greatest element would be one that all the other elements divide. The only two candidates (maximal elements) are 24 and 45, and since neither divides the other, we conclude that there is no greatest element.

15. A least element would be one that divides all the other elements. The only two candidates (minimal elements) are 3 and 5, and since neither divides the other, we conclude that there is no least element.

Further Reading

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. *Discrete mathematics with applications*. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit- 04: Boolean Algebra

CONTENTS

Objectives/Expected Learning Outcome

Introduction

Self-Assessment

4.1 Identities of Boolean Algebra

Self-Assessment

4.2 The Abstract Definition of a Boolean Algebra

Self-Assessment

Summary

Answers of Self-Assessment

Further/ Suggested Readings

Objectives/Expected Learning Outcome

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are: -

- understand what are Boolean Functions
- understand what are Boolean Expressions
- understand how to write Boolean Expressions into a logical equivalence.
- understand what are duals of Boolean expression.
- understand what is a Complemented Lattice.
- understand what is Boolean Algebra.

Introduction

Boolean algebra provides the operations and the rules for working with the set{0,1}. Electronic and optical switches can be studied using this set and the rules of Boolean algebra. The three operations in Boolean algebra that we will use most are complementation, the Boolean sum, and the Boolean product. The complement of an element, denoted with a bar, is defined by $\bar{0} = 1$ and $\bar{1} = 0$. The Boolean sum, denoted by+or by OR, has the following values: 1+1 = 1, 1+0 = 1, 0+1 = 1, 0+0 = 0. The Boolean product, denoted by ·or by AND, has the following values: 1 ·1 = 1, 1 ·0 = 0, 0 ·1 = 0, 0 ·0 = 0. When there is no danger of confusion, the symbol ·can be deleted, just as in writing algebraic products. Unless parentheses are used, the rules of precedence for Boolean operators are: first, allcomplementsarecomputed,followedbyallBooleanproducts,followedbyallBooleansums. This is illustrated in Example 1.

Example 1 Find the value of $1 \cdot 0+(\bar{1})$.

Solution: Using the definitions of complementation, the Boolean sum, and the Boolean product, it follows that $1 \cdot 0+(+(\bar{1}).) = 0+1 = 0+0 = 0$.

Self-Assessment

1. Find the values of the expression $1 \cdot \bar{0}$.
2. Find the values of the expression $1+\bar{1}$.

3. Find the values of expression

4.Find the values of expression (0).

5. Show that (1 1)+(0) +0) = 1.

6. Translate the equation in 2 into a propositional equivalence by changing each 0 into an F, each 1 into a T, each Boolean sum into a disjunction, each Boolean product into a conjunction, each complementation into a negation, and the equals sign into a propositional equivalence sign.

7. What values of the Boolean variables x and y satisfy xy = x +y?

The complement, Boolean sum, and Boolean product correspond to the logical operators, ¬,∨, and∧, respectively, where 0 corresponds to F (false) and 1 corresponds toT (true). Equalities in Boolean algebra can be directly translated into equivalences of compound propositions. Conversely, equivalences of compound propositions can be translated into equalities in Boolean algebra. We will see later in this section why these translations yield valid logical equivalences and identities in Boolean algebra. Example 2 illustrates the translation from Boolean algebra to propositional logic.

Example 2 Translate 1 0+(the equality found in Example 1, into a logical equivalence.

Solution: We obtain a logical equivalence when we translate each 1 into a T, each 0 into an F, each Boolean sum into a disjunction, each Boolean product into conjunction, and each complementation into a negation. We obtain

(T∧F)∨¬ (T∨F) ≡ F.▲

Example 3 illustrates the translation from propositional logic to Boolean algebra.

Example 3 Translate the logical equivalence (T∧T)∨¬F ≡ T into an identity in Boolean algebra.

Solution: We obtain identity in Boolean algebra when we translate each T into a 1, each F into a 0, each disjunction into a Boolean sum, each conjunction into a Boolean product, and each negation into complementation. We obtain

$$(1 \cdot 1)+\bar{0}) = 1.$$

Boolean Expressions and Boolean Functions Let B ={ 0,1}. Then B^n ={$(x_1,x_2,...,x_n)$ | xi ∈ B for 1 ≤i≤ n} is the set of all possiblen -tuples of 0s and 1s. The variable x is called a Boolean variable if it assumes values only from B, that is, if its only possible values are 0 and 1. A function from Bn to B is called a Boolean function of degree n.

Example 4 The function F(x, y)= xy from the set of ordered pairs of Boolean variables to the set{0,1}is a Boolean function of degree 2 with F(1,1) = 0, F(1,0) = 1, F(0,1) = 0, and F(0,0) = 0. We display these values of F in Table 1.

Table1

x	Y	F(x,y)
1	1	0
1	0	1
0	1	0
0	0	0

Booleanfunctionscanberepresentedusingexpressionsmadeupfromariablesand Boolean operations. The Boolean expressions in the variables $x_1,x_2,...,x_n$ are define recursively as $0,1,x_1,x_2,...,x_n$ are Boolean expressions; if E_1 and E_2 are Boolean expressions, then $\bar{E}_1$(E_1E_2), and (E_1 +E_2) are Boolean expressions.

EachBooleanexpressionrepresentsaBooleanfunction.Thevaluesofthisfunctionareobtained by substituting 0 and 1 for the variables in the expression. In Section 12.2 we will show that every Boolean function can be represented by a Boolean expression.

Example 5 Find the values of the Boolean function represented by F(x, y, z)= xy+ Solution: The values of this function are displayed in Table 2.

Solution: The values of this function are displayed in Table 2.

Table2

x	y	z	xy		F(x,y,z)=xy+
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1

Note that we can represent a Boolean function graphically by distinguishing the vertices of the n-cube that correspond to the n-tuples of bits where the function has value 1.

Example 6 The function F(x, y, z)= xy+ z¯from B3 to B from Example 5 can be represented by distinguishing the vertices that correspond to the five 3-tuples (1,1,1),(1,1,0), (1,0,0), (0,1,0), and (0,0,0), where F(x, y, z)= 1, as shown in Figure 1. These vertices are displayed using solid black circles.

$(1,b_2,...,b_n)$ whenever $b_1,b_2,...,b_n$ belong to B.Two different Boolean expressions that represent the same function are called equivalent. For instance, the Boolean expressions xy, xy+0, and xy ·1 are equivalent.The complement of the Boolean function F is the function F, where $F(x_1,...,x_n) = F(x_1,...,x_n)$. Let F and G be Boolean functions of degree n. The Boolean sum F +G and the Boolean product FGare defined by $(F+G)(x_1,...,x_n) = F(x_1,...,x_n)+G(x_1,...,x_n)$, $(FG)(x_1,...,x_n) = F(x_1,...,x_n)G(x_1,...,x_n)$. A Boolean function of degree two is a function from a set with four elements, namely, pairs of elements from B ={ 0,1}, toB, a set with two elements. Hence, there are 16 different Boolean functions of degree two. In Table 3 we display the values of the 16 different Boolean functions of degree two, labelled $F_1,F_2,...,F_{16}$.

Table 3 The 16 Boolean Functions of Degree Two

x	y	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0

Example 7 How many different Boolean functions of degree n are there?

Solution: From the product rule for counting, it follows that there are 2^n different n-tuples of 0s and 1s. Because a Boolean function is an assignment of 0 or 1 to each of these 2^n different n-tuples, the product rule shows that there are 2^{2^n} different Boolean functions of degree n. ▲

Table 4 displays the number of different Boolean functions of degrees one through six. The number of such functions grows extremely rapidly.

Table 4 The Number of Boolean Functions of Degree n.	
Degree	Number
1	4
2	16
3	256
4	65536
5	4,294,967,296
6	18,446,744,073,709,551,616

4.1 Identities of Boolean Algebra

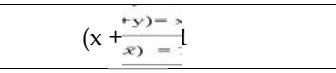
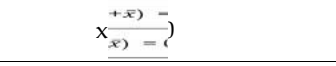
There are many identities in Boolean algebra. The most important of these are displayed in Table 5. These identities are particularly useful in simplifying the design of circuits. Each of the identities in Table 5 can be proved using a table. We will prove one of the distributive laws in this way in Example 8. The proofs of the remaining properties are left as exercises for the reader.

Example 8 Show that the distributive law x(y+z)=xy+xz is valid. Solution: The verification of this identity is shown in Table 6. The identity holds because the last two columns of the table agree.

▲

The reader should compare the Boolean identities in Table 5 to the logical equivalences in Table 6 of Section 1.3 and the set identities in Table 1 in Section 2.2. All are special cases of the same set of identities in a more abstract structure. Each collection of identities can be obtained by making the appropriate translations. For example, we can transform each of the identities in Table 5 into a logical equivalence by changing each Boolean variable into a propositional variable,each 0into an F,each1intoaT,eachBooleansumintoadisjunction,each Boolean product into conjunction, and each complementation into a negation, as we illustrate in Example 9.

Table 5 Boolean Identities.	
Identity	Name
$\overline{\bar{x}} = x$	Law of the double complement
x +x = x x ·x = x	Idempotent laws
x +1 = 1 x ·0 =	Domination laws
0 x +y = y +x xy = yx	Commutative laws
x +(y +z)= (x +y)+z x(yz)= (xy)z	Associative laws

x +yz= (x +y)(x +z) x(y+z)=xy+xz	Distributive laws
	De Morgan's laws
x +xy = x x(x+y)= x	Absorption laws
	Unit property
	Zero property

Self-Assessment

8. Verify the idempotent laws.
9. Verify the domination laws.
10. Verify the associative laws.
11. Verify De Morgan's laws.

Example 9 Translate the distributive law x +yz= (x +y)(x +z) in Table 5 into a logical equivalence.

Solution: To translate a Boolean identity into a logical equivalence, we change each Boolean variable into a propositional variable. Here we will change the Boolean variables x, y, and z into thepropositionalvariablesp,q,andr.Next,wechangeeachBooleansumintoadisjunctionand each Boolean product into conjunction. (Note that 0 and 1 do not appear in this identity and

Table 6 Verifying One of the Distributive Laws.

x	y	z	y+z	xy	xz	x(y+z)	xy+xz
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

complementation also does not appear.) This transforms the Boolean identity into the logical equivalence

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r).$$

This logical equivalence is one of the distributive laws for propositional logic in Table 6 in Section 1.3. ▲

Identities in Boolean algebra can be used to prove further identities. We demonstrate this in Example 10.

Example 10 Prove the absorption law x(x+y)= x using the other identities of Boolean algebra shown in Table 5. (This is called an absorption law because absorbing x +y into x leaves x unchanged.)

Solution: We display steps used to derive this identity and the law used in each step:

x(x+y)= (x +0)(x +y) Identity law for the Boolean sum = x +0 ·y Distributive law of the Boolean sum over the Boolean product = x +y ·0 Commutative law for the Boolean product = x +0 Domination law for the Boolean productDuality

The identities inTable 5 come in pairs (except for the law of the double complement and the unit and zero properties). To explain the relationship between the two identities in each pair we use the concept of a dual. The dual of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0s and 1s.

Example 11 Find the duals of x(x+0) and $x\cdot 1+(y+z)$.

Solution: Interchanging · signs and + signs and interchanging 0s and 1s in these expressions produces their duals. The duals are x +(y ·1) and $(\bar{x}+0)(\bar{y}\bar{z})$, respectively0.

The dual of a Boolean function F represented by a Boolean expression is the function representedbythedualofthisexpression.ThisdualfunctionF, denotedbyF^d, doesnotdependon theparticularBooleanexpressionusedtorepresentF. AnidentitybetweenfunctionsrepresentedbyBoolean expressions remains valid when the duals of both sides of the identity are taken. This result, called the duality principle, is useful for obtaining new identities.

Example 12 Construct an identity from the absorption law x(x+y)= x by taking duals.

Solution:Taking the duals of both sides of this identity produces the identity x+xy = x, which is also called an absorption law and is shown in Table 5.

4.2 The Abstract Definition of a Boolean Algebra

In this section, we have focused on Boolean functions and expressions. However, the results we have established can be translated into results about propositions or results about sets. Because of this, it is useful to define Boolean algebras abstractly. Once it is shown that a particular structure is a Boolean algebra, then all results established about Boolean algebras, in general, apply to this particular structure. Boolean algebras can be defined in several ways. The most common way is to specify the properties that operations must satisfy, as is done in Definition 1.

DEFINITION 1 A Boolean algebra is a set B with two binary operations ∨ and ∧, elements 0 and 1, and a unary operation such that these properties hold for all x, y, and z in B:

$\left\{\begin{matrix} x \wedge x = 0 \\ x \wedge 1 = x \end{matrix}\right\}$ Identity laws

$x \vee \bar{x} = 1$ $x \wedge \bar{x} = 0$ Complement laws

{x ∨y)∨z = x ∨(y ∨z}

{x ∧y)∧z = x ∧(y ∧z} Associative laws

x ∨y = y ∨x x∧y = y ∧x} Commutative laws

x ∨(y ∧z)= (x ∨y)∧(x ∨z)}

x ∧(y ∨z)= (x ∧y)∨(x ∧z) }Distributive laws

Using the laws given in Definition 1, it is possible to prove many other laws that hold for every Boolean algebra, such as idempotent and domination laws. From our previous discussion, B ={ 0,1} with the OR and AND operations and the complement operator, satisfies all these properties. The set of propositions in n variables, with the∨ and∧operators, F and T, and the negation operator, also satisfies all the properties of a Boolean algebra, as can be seen from Table 6. Similarly, the set of subsets of a universal set U with the union and intersection operations, the empty set and the universal set, and the set complementation operator, is a Boolean algebra as can be seen by consulting Table 1 in Section 2.2. So, to establish results about each of the Boolean expressions, propositions, and sets, we need only prove results about abstract Boolean algebras. Boolean algebras may also be defined using the notion of a lattice, discussed in Chapter 9. Recall that a lattice L is a partially ordered set in which every pair of elements x,y has a least upper bound, denoted by lub(x,y) and a greatest lower bound denoted by glb(x,y). Given two elements x and y of

L, we can define two operations ∨and ∧ on pairs of elements of L by x ∨y = lub(x,y) and x ∧y =glb(x,y). ForalatticeLtobeaBooleanalgebraasspecifiedinDefinition1,itmusthavetwoproperties. First, it must be complemented. For a lattice to be complemented it must have a least element 0 and the greatest element 1 and for every element x of the lattice, there must exist an element x such that x ∨x = 1 and x ∧x = 0. Second, it must be distributive. This means that for every x, y, and z in L, x ∨(y ∧z)= (x ∨y)∧(x ∨z) and x ∧(y ∨z)= (x ∧y)∨(x ∧z).

Self-Assessment

12. Show that in a Boolean algebra, the idempotent laws x ∨x = x and x ∧x = x hold for every element x.

13. Show that in a Boolean algebra, if x ∨y = 0, then x = 0 and y = 0, and that if x ∧y = 1, then x = 1 and y = 1.

14. Show that a complemented, distributive lattice is a Boolean algebra.

Summary

The key concepts learned from this unit are: -

- We have learned what are Boolean Functions
- We have learned what are Boolean Expressions
- We have learned how to write Boolean Expressions into a logical equivalence.
- We have learned what are duals of Boolean expression.
- We have learned what is a Complemented Lattice.

We have learned what is Boolean Algebra.

Answers of Self-Assessment

1.

$1 \cdot \bar{0} = 1 \cdot 1 = 1$

2.

$1+\bar{1}=1+0=1$

3.

$\bar{0} \cdot 0=0$

4.

$(\bar{\bar{0}}) =1= 0$

5.

We compute $(1 \cdot 1) + ((1.0)\bar{}\ ) = 1+((0+0)\bar{}\ ) = 1+(1+0) = 1+1=1$.

6. Following the instructions, we have (T ∧T) V (•(F∧ T) ∨F) = T.

7.By looking at the definitions, we see that this equation is satisfied if and only if x = y, i.e., x = y = 0 or x=y=1.

8.The idempotent laws state that x · x = x and x + x = x. There are only four things to check: 0 · 0 = 0, 0 + 0 = 0, 1 · 1 = 1, and 1 + 1 = 1, all of which are part of the definitions. The relevant tables, exhibiting these calculations, have only two rows.

9.The domination laws state that x + 1 = 1 and x · 0 = 0. There are only four things to check: 0 + 1 = 1, 0 · 0 = 0, 1 + 1 = 1, and 1 · 0 = 0, all of which are part of the definitions. The relevant tables, exhibiting these calculations, have only two rows.

10.We can verify each associative law by constructing the relevant table, which will have eight rows, since there are eight combinations of values for x, y, and z in the equations x + (y + z) = (x +

y) + z and x(yz) = (xy)z. Rather than write down these tables, let us observe that in the first case, both sides are equal to 1 unless x = y = z = 0 (in which case both sides equal 0), and, dually, in the second case, both sides are equal to 0 unless x = y = z = 1 (in which case both sides equal 1).

11.We construct relevant tables and compute the quantities shown. Since the fourth and seventh columns are identical we conclude that $\overline{(x+y)}=\bar{x}+\bar{y}$;since the ninth and tenth columns are equal, we conclude that $\overline{(xy)} = \bar{x}\bar{y}$

x	y	xy	$\overline{(xy)}$	$\bar{x}$	$\bar{y}$	$\bar{x}+\bar{y}$	x+y	$\overline{(x+y)}$	$\bar{x}\bar{y}$
1	1	1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0
0	0	0	1	1	1	1	0	1	1

12.We need to play around with the symbols until the desired results fall out. To prove that x V x = x, let us compute x V (x ∧ $\bar{x}$) in two ways. On the one hand,

x∨ (x ∧ $\bar{x}$) =x∨O = x

by the complement law and the identity law. On the other hand, by using the distributive law followed by the complement and identity laws we have

x∨ (x ∧ $\bar{x}$) =(x∨ x) ∧ (x∨ $\bar{x}$) = (x V x) ∧ 1 = x V x.

By transitivity of equality, x = x V x. The other property is the dual of this one, and its proof can be obtained by formally replacing every V with ∧ and replacing every 0 with 1 , and vice versa. Thus our proof, shortened to one line, becomes

x = x ∧ 1 = x ∧(x V $\bar{x}$) = (x ∧ x) V (x ∧ x) = (x ∧ x) V 0 = x V x.

13. Using the hypothesis, we compute as follows. x = x V 0 = x V (x Vy) = (x V x) Vy= x Vy= 0. Similarly for y, by the commutative law. Note that we used the result of Exercise 35. The other statement is proved in the dual manner: x = x ∧ 1 = x ∧ (x ∧ y) = (x ∧ x) ∧ y = x ∧y = 1 and similarly for y.

14.We need to verify the five laws (each one consisting of a dual pair, of course). The complement laws are part of the definition of "complemented". Finally, the distributive laws are again part of the definition.

Further/ Suggested Readings

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. *Discrete mathematics with applications*. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 05: Introduction, Basic Counting Principles

CONTENTS

Objectives
Introduction
5.1 Counting
5.2 Permutations and Combinations
Summary
Self Assessment
Answer for Self Assessment
Review Questions
Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- understand what are the basics of counting.
- understand the use of product rule in the basics of counting.
- understand the use of Sum rule in the basics of counting.
- understand the use of the Subtraction rule in the basics of counting.
- understand the use of tree diagram in the basics of counting.
- understand the Division rule in the basics of counting.
- understand how to find the total number of functions in the basics of counting.

Introduction

Suppose that a password on a computer system consists of six, seven, or eight characters. Each of these characters must be a digit or a letter of the alphabet. Each password must contain at least 1 one digit. How many such passwords are there? The techniques needed to answer this question and a wide variety of other counting problems will be introduced in this section.

Counting problems arise throughout mathematics and computer science. For example, we must count the successful outcomes of experiments and all the possible outcomes of these experiments to determine the probabilities of discrete events. We need to count the number of operations used by an algorithm to study its time complexity.

We will introduce the basic techniques of counting in this section. These methods serve as the foundation for almost all counting techniques. We first present two basic counting principles, the product rule and the sum rule. Then we will show how they can be used to solve many different counting problems. The product rule applies when a procedure is made up of separate tasks.

5.1 Counting

Combinatorics, the study of arrangements of objects, is an important part of discrete mathematics. This subject was studied as long ago as the seventeenth century when combinatorial questions arose in the study of gambling games. Enumeration, the counting of objects with certain properties, is an important part of combinatorics. We must count objects to solve many different types of problems. For instance, counting is used to determine the complexity of algorithms. Counting is also required to determine whether there are enough telephone numbers or Internet protocol addresses to meet demand. Recently, it has played a key role in mathematical biology, especially in sequencing DNA. Furthermore, counting techniques are used extensively when probabilities of events are computed.

The basic rules of counting, which we will study in later, can solve a tremendous variety of problems. For instance, we can use these rules to enumerate the different telephone numbers possible in the United States, the allowable passwords on a computer system, and the different orders in which the runners in a race can finish. Another important combinatorial tool is the pigeonhole principle, which we will study in the coming section. This states that when objects are placed in boxes and there are more objects than boxes, then there is a box containing at least two objects. For instance, we can use this principle to show that among a set of 15 or more students, at least 3 were born on the same day of the week.

We can phrase many counting problems in terms of ordered or unordered arrangements of the objects of a set with or without repetitions. These arrangements, called permutations and combinations, are used in many counting problems. For instance, suppose the 100 top finishers on a competitive exam taken by 2000 students are invited to a banquet. We can count the possible sets of 100 students that will be invited, as well as how the top 10 prizes can be awarded.

Another problem in combinatorics involves generating all the arrangements of a specified kind. This is often important in computer simulations. We will devise algorithms to generate arrangements of various types.

The product rule

Suppose that a procedure can be broken down into a sequence of two tasks. If there are n1 ways to do the first task and for each of these ways of doing the first task, there are n2 ways to do the second task, then there are n1n2 ways to do the procedure. Example s 1–10 show how the product rule is used.

1. A new company with just two employees, Sanchez and Patel, rents a floor of a building with12 offices. How many ways are there to assign different offices to these two employees?

Solution: The procedure of assigning offices to these two employees consists of assigning an office to Sanchez, which can be done in 12 ways, then assigning an office to Patel different from the officer assigned to Sanchez, which can be done in 11 ways. By the product rule, there are $12 \cdot 11 = 132$ ways to assign offices to these two employees. ▲

2. The chairs of an auditorium are to be labelled with an uppercase English letter followed by a positive integer not exceeding 100. What is the largest number of chairs that can be labelled differently?

Solution:

The procedure of labelling a chair consists of two tasks, namely, assigning to the seat one of the 26 uppercase English letters, and then assigning to it one of the 100 possible integers.

The product rule shows that there are $26 \cdot 100 = 2600$ different ways that a chair can be labelled. Therefore, the largest number of chairs that can be labelled differently is 2600. ▲

3. There are 32 microcomputers in the computer centre. Each microcomputer has 24 ports. How

many different ports to a microcomputer in the centre are there?

Solution:

The procedure of choosing a port consists of two tasks, first picking a microcomputer and then picking a port on this microcomputer. Because there are 32 ways to choose the micro-computer and 24 ways to choose the port no matter which microcomputer has been selected, the product rule shows that there are $32 \cdot 24 = 768$ ports. ▲

An extended version of the product rule is often useful. Suppose that a procedure is carried out by performing the tasks $T_1, T_2,...,T_m$ in sequence. If each task T_i, i = 1, 2,...,n, can be done in ni ways, regardless of how the previous tasks were done, then there are $n_1 \cdot n_2 \cdot \cdots \cdot nm$ ways to carry out the procedure. This version of the product rule can be proved by mathematical induction from the product rule for two tasks.

4. How many different bit strings of length seven are there?

Solution:

Each of the seven bits can be chosen in two ways because each bit is either 0 or 1. Therefore, the product rule shows there are a total of $2^7 = 128$ different bit strings of length seven. ▲

5. How many different license plates can be made if each plate contains a sequence of three uppercase English letters followed by three digits (and no sequences of letters are prohibited, even if they are obscene)?

---- ---- ---	----------
26 choices	10 choices
for each	for each
letter	digit

Solution:

There are 26 choices for each of the three uppercase English letters and ten choices for each of the three digits. Hence, by the product rule there are a total of $26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17{,}576{,}000$ possible license plates. ▲

6. Counting Functions How many functions are there from a set with m elements to a set with

n elements?

Solution:

A function corresponds to a choice of one of the n elements in the codomain for each of the elements in the domain. Hence, by the product rule, there are $n \cdot n \cdot \cdots \cdot n = n^m$ functions from a set with m elements to one with n elements. For example, there are $5^3 = 125$ different functions from a set with three elements to a set with five elements. ▲

7. Counting One-to-One Functions How many one-to-one functions are there from a set with

m elements to one with n elements?

Solution:

First, note that when m>n there are no one-to-one functions from a set with m

elements to a set with n elements to a set with n elements.

Now let $m \leq n$. Suppose the elements in the domain are $a_1, a_2, \ldots, a_m$. There are n ways to choose the value of the function at a1. Because the function is one-to-one, the value of the function at a_2 can be picked in $n - 1$ way (because the value used for a_1 cannot be used again).

In general, the value of the function at a_k can be chosen in $n - k + 1$ ways. By the product rule, there are $n(n - 1)(n - 2) \cdots (n - m + 1)$ one-to-one functions from a set with m elements to one with n elements. For example, there are $5 \cdot 4 \cdot 3 = 60$ one-to-one functions from a set with three elements to a set with five elements.

More Complex Counting Problems

Many counting problems cannot be solved using just the sum rule or just the product rule. However, many complicated counting problems can be solved using both of these rules in combination. We begin by counting the number of variable names in the programming language BASIC. (In the exercises, we consider the number of variable names in JAVA.) Then we will count the number of valid passwords subject to a particular set of restrictions.

15. In a version of the computer language BASIC, the name of a variable is a string of one or two alphanumeric characters, where uppercase and lowercase letters are not distinguished. (An alphanumeric character is either one of the 26 English letters or one of the 10 digits.) Moreover, a variable name must begin with a letter and must be different from the five strings of two characters that are reserved for programming use. How many different variable names are there in this version of BASIC?

Solution:

Let V equal the number of different variable names in this version of BASIC. Let v_1 be the number of these that are one character long and v_2 be the number of these that are two characters long. Then by the sum rule, V = v_1 + v_2. Note that v_1 = 26, because a one-character variable name must be a letter. Furthermore, by the product rule, there are 26 · 36 strings of length two that begin with a letter and end with an alphanumeric character. However, five of these are excluded, so v_2 = 26 · 36 − 5 = 931. Hence, there are V = v_1 + v_2 = 26 + 931 = 957 different names for variables in this version of BASIC.

16. Each user on a computer system has a password, which is six to eight characters long, where

each character is an uppercase letter or a digit. Each password must contain at least one digit. How many possible passwords are there?

Solution:

Let P be the total number of possible passwords, and let p_6, p_7, and p_8, denote the number of possible passwords of length 6, 7, and 8, respectively. By the sum rule,

P = P_6 + P_7 + P_8 We will now find P_6 + P_7 and P_8. Finding P_6 directly is difficult. To find P_6 it is easier to find the number of strings of uppercase letters and digits that are six characters long, including those with no digits, and subtract from this the number of strings with no digits. By the product rule, the number of strings of six characters is 366, and the number of strings with no digits is 266. Hence,

P_6 = 36^6 − 26^6 = 2,176,782,336 − 308,915,776 = 1,867,866,560. Similarly, we have P_7 = 36^7 − 26^7 = 78,364,164,096 − 8,031,810,176 = 70,332,353,920 and P_8 = 36^8 − 26^8 = 2,821,109,907,456 − 208,827,064,576 = 2,612,282,842,880.

Consequently, P = P_6 + P_7 + P_8 = 2,684,483,063,360.▲

17.

Counting Internet Addresses On the Internet, which is made up of interconnected physical networks of computers, each computer (or more precisely, each network connection of a computer) is assigned an Internet address. In Version 4 of the Internet Protocol (IPv4), now in use,

Bit number	0	1	2	3	4	8	16	24	31
Class A	0	netid				hostid			
Class B	1	0	netid				hostid		
Class C	1	1	0	netid				hostid	
Class D	1	1	1	0	Multicast address				
Class E	1	1	1	1	0	address			

.an address is a string of 32 bits. It begins with a network number (netid). The netid is followed by a host number (hostid), which identifies a computer as a member of a particular network. Three forms of addresses are used, with different numbers of bits used for netids and hostids. Class A addresses, used for the largest networks, consist of 0, followed by a 7-bit netid and a 24-bit hostid.

Class B addresses, used for medium-sized networks, consist of 10, followed by a 14-bit netid and a 16-bit hostid. Class C addresses, used for the smallest networks, consist of 110, followed by a 21-bit netid and an 8-bit hostid. There are several restrictions on addresses because of special uses: 1111111 is not available as the netid of a Class A network, and the hostids consisting of all 0s and all 1s are not available for use in any network. A computer on the Internet has either a Class A, a Class B, or a Class C address. (Besides Class A, B, and C addresses, there are also Class D addresses, reserved for use in multicasting when multiple computers are addressed at a single time, consisting of 1110 followed by 28 bits, and Class E addresses, reserved for future use, consisting of 11110 followed by 27 bits. Neither Class D nor Class E addresses are assigned as the IPv4 address of a computer on the Internet.)

Figure 1 illustrates the IPv4 addressing. (Limitations on the number of Class A and Class B netids have made IPv4 addressing inadequate; IPv6, a new version of IP, uses 128-bit addresses to solve this problem.)

How many different IPv4 addresses are available for computers on the Internet?

Solution:

Let x be the number of available addresses for computers on the Internet, and let andx_A, x_B,x_c denote the number of Class A, Class B, and Class C addresses available, respectively.

By the sum rule, x =x_A+ $x_B + x_c$

To find x_A, note that there are 27 − 1 = 127 Class A netids, recalling that the netid 1111111 is unavailable. For each netid, there are 224 − 2 = 16,777,214 hostids, recalling that the hostids consisting of all 0s and all 1s are unavailable. Consequently, xA = 127 ·16,777,214 = 2,130,706,178.

To find x_A and x_c, note that there are 214 = 16,384 Class B netids and 221 = 2,097,152 Class C netids. For each Class B netid, there are 216 − 2 = 65,534 hostids, and for each Class C netid, there are 28 − 2 = 254 hostids, recalling that in each network the hostids consisting of all 0s and all 1s are unavailable. Consequently, xB = 1,073,709,056 and xC = 532,676,608. We conclude that the total number of IPv4 addresses available is x = $x_A + x_B + x_C$= 2,130,706,178 + 1,073,709,056 + 532,676,608 = 3,737,091,842.

18. How many bit strings of length eight either start with a 1 bit or end with the two bits 00?

Solution:

We can construct a bit string of length eight that either starts with a 1 bit or ends with the two bits 00, by constructing a bit string of length eight beginning with a 1 bit or by constructing a bit string of length eight that ends with the two bits 00. We can construct a bit string of length eight that begins with a 1 in 2^7 = 128 ways. This follows by the product rule because the first bit can be chosen in only one way and each of the other seven bits can be chosen in two ways. Similarly, we can construct a bit string of length eight ending with the two bits 00, in 2^6 = 64 ways. This follows by the product rule because each of the first six bits can be chosen in two ways and the last

two bits can be chosen in only one way.

1	2^7=128 way		
0	2^6 =64 ways	0	0
1 0	2^5=32 ways		0

Some of the ways to construct a bit string of length eight starting with a 1 are the same as the ways to construct a bit string of length eight that ends with the two bits 00. There are 25 = 32 ways to construct such a string. This follows by the product rule because the first bit can be chosen in only one way, each of the seconds through the sixth bits can be chosen in two ways, and the last two bits can be chosen in one way. Consequently, the number of bit strings of length eight that begin with a 1 or end with a 00, which equals the number of ways to construct a bit string of length eight that begins with a 1 or that ends with 00, equals 128 + 64 − 32 = 160. We present an example that illustrates how the formulation of the principle of inclusion-exclusion can be used to solve counting problems.

19. A computer company receives 350 applications from computer graduates for a job planning a line of new Web servers. Suppose that 220 of these applicants majored in computer science, 147 majored in business, and 51 majored both in computer science and in business. How many of these applicants majored neither in computer science nor in business?

Solution:

To find the number of these applicants who majored neither in computer science nor in business, we can subtract the number of students who majored either in computer science or in business (or both) from the total number of applicants. Let A1 be the set of students who majored in computer science and A2 the set of students who majored in business. Then $A_1 \cup A_2$ is the set of students who majored in computer science or business (or both), and $A_1 \cap A_2$ is the set of students who majored both in computer science and in business. By the subtraction rule, the number of students who majored either in computer science or in business (or both) equals

$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$ = 220 + 147 − 51 = 316. We conclude that 350 − 316 = 34 of the applicants majored neither in computer science nor in business. ▲ The subtraction rule, or the principle of inclusion-exclusion, can be generalized to find the number of ways to do one of n different tasks or, equivalently, to find the number of elements in the union of n sets, whenever n is a positive integer. We will study the inclusion-exclusion principle and some of its many applications in Chapter 8.

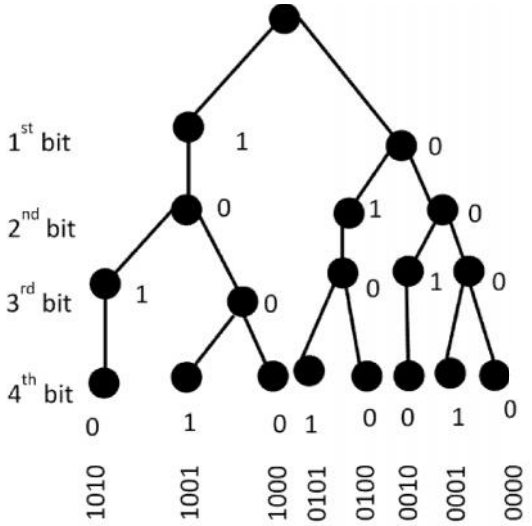


FIGURE 2 Bit strings of length four without consecutive 1's

Tree Diagrams

Counting problems can be solved using tree diagrams. A tree consists of a root, several branches leaving the root, and possible additional branches leaving the endpoints of other branches. To use trees in counting, we use a branch to represent each possible choice. We represent the possible outcomes by the leaves, which are the endpoints of branches not having other branches starting at them.

Note that when a tree diagram is used to solve a counting problem, the number of choices

of which branch to follow to reach a leaf can vary.

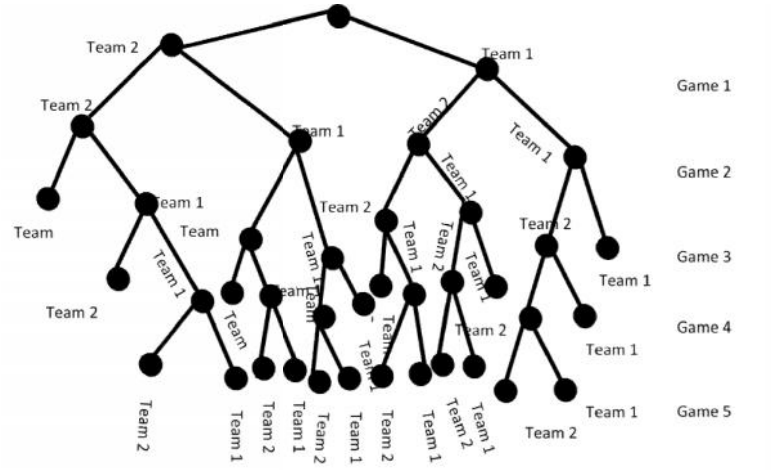


FIGURE 4. Best three games out of five playsoff

21. How many bit strings of length four do not have two consecutive 1s?

Solution:

The tree diagram in Figure 2 displays all bit strings of length four without two consecutive 1s. We see that there are eight-bit strings of length four without two consecutive 1s. ▲

22. A playoff between two teams consists of at most five games. The first team that wins three games wins the playoff. In how many different ways can the playoff occur?

Solution:

The tree diagram in Figure 3 displays all the ways the playoff can proceed, with the winner of each game shown. We see that there are 20 different ways for the playoff to occur. ▲

23. Suppose that "I Love New Jersey" T-shirts come in five different sizes: S, M, L, XL, and XXL.

Further suppose that each size comes in four colours, white, red, green, and black, except for XL, which comes only in red, green, and black, and XXL, which comes only in green and black. How many different shirts do a souvenir shop have to stock to have at least one of each available size and colour of the T-shirt?

Solution:

The tree diagram in Figure 4 displays all possible size and colour pairs. It follows that the souvenir shop owner needs to stock 17 different T-shirts.

5.2 Permutations and Combinations

Introduction Many counting problems can be solved by finding the number of ways to arrange a specified number of distinct elements of a set of a particular size, where the order of these elements matters. Many other counting problems can be solved by finding the number of ways to select a particular number of elements from a set of a particular size, where the order of the elements selected does not matter. For example, in how many ways can we select three students from a group of five students to stand in line for a picture? How many different committees of three students can be formed from a group of four students? In this section, we will develop methods to answer questions such as these. Permutations We begin by solving the first question posed in the introduction to this section, as well as related questions.

1. In how many ways can we select three students from a group of five students to stand in line for a picture? In how many ways can we arrange all five of these students in a line for a picture?

Solution: First, note that the order in which we select the student's matters. There are five ways to select the first student to stand at the start of the line. Once this student has been selected, there are four ways to select the second student in the line. After the first and second students have been selected, there are three ways to select the third student in the line. By the product rule, there are $5 \cdot 4 \cdot 3 = 60$ ways to select three students from a group of five students to stand in line for a picture. To arrange all five students in a line for a picture, we select the first student in five ways, the second in four ways, the third in three ways, the fourth in two ways, and the fifth in one way. Consequently, there are $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ ways to arrange all five students in a line for a picture.

2. Let S = {1, 2, 3}. The ordered arrangement 3, 1, 2 is a permutation of S. The ordered arrangement 3, 2 is a 2-permutation of S. ▲ The number of r-permutations of a set with n elements is denoted by P (n, r). We can find P (n, r) using the product rule.

3. Let S = {a, b, c}. The 2-permutations of S are the ordered arrangements a, b; a, c; b, a; b, c; c, a; and c, b. Consequently, there are six 2-permutations of this set with three elements. There are always six 2-permutations of a set with three elements. There are three ways to choose the first element of the arrangement. There are two ways to choose the second element of the arrangement because it must be different from the first element. Hence, by the product rule, we see that $P(3, 2) = 3 \cdot 2 = 6$. the first element. By the product rule, it follows that $P(3, 2) = 3 \cdot 2 = 6$.

We now use the product rule to find a formula for P (n, r) whenever n and r are positive integers with $1 \le r \le n$.

THEOREM 1

If n is a positive integer and r is an integer with $1 \le r \le n$, then there are $P(n, r) = n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of a set with n distinct elements. Proof: We will use the product rule to prove that this formula is correct. The first element of the permutation can be chosen in n ways because there are n elements in the set. There are $n-1$ ways to choose the second element of the permutation because there are $n-1$ elements left in the set after using the element picked for the first position. Similarly, there are $n-2$ ways to choose the third element, and so on, until there are exactly $n-(r-1) = n-r+1$ ways to choose the r^{th} element. Consequently, by the product rule, there are $n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of the set. Note that $P(n, 0) = 1$ whenever n is a non-negative integer because there is exactly one way to order zero elements. That is, there is exactly one list with no elements in it, namely the empty list. We now state a useful corollary of Theorem 1. COROLLARY 1

If n and r are integers with $0 \le r \le n$, then $P(n, r) = \frac{n!}{(n-r)!}$. Proof: When n and r are integers with $1 \le r \le n$, by Theorem 1 we have $P(n, r) = n(n-1)(n-2)\cdots\frac{n!}{(n-r)!}$. Because $\frac{n!}{(n-0)!} = \frac{n!}{n!} = 1$ whenever n is a non-negative integer, we see that the formula $P(n, r) = \frac{n!}{(n-r)!}$. also holds when $r = 0$.

By Theorem 1 we know that if n is a positive integer, then $P(n, n) = n!$. We will illustrate this result with some examples.

4. How many ways are there to select a first-prize winner, a second-prize winner, and a third-prize winner from 100 different people who have entered a contest?

Solution:

Because it matters which person wins which prize, the number of ways to pick the three prize winners is the number of ordered selections of three elements from a set of 100 elements, that is, the number of 3-permutations of a set of 100 elements. Consequently, the answer is $P(100, 3) = 100 \cdot 99 \cdot 98 = 970{,}200$. ▲

5. Suppose that there are eight runners in a race. The winner receives a gold medal, the second-place finisher receives a silver medal, and the third-place finisher receives a bronze medal. How many different ways are there to award these medals, if all possible outcomes

of the race can occur and there are no ties?

Solution:

The number of different ways to award the medals is the number of 3-permutations of a set with eight elements. Hence, there are P (8, 3) = 8 ·7 ·6 = 336 possible ways to award the medals. ▲

6. Suppose that a saleswoman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities?

Solution:

The number of possible paths between the cities is the number of permutations of seven elements because the first city is determined, but the remaining seven can be ordered arbitrarily. Consequently, there are 7! = 7 ·6 ·5 ·4 ·3 ·2 ·1 = 5040 ways for the saleswoman to choose her tour. If, for instance, the saleswoman wishes to find the path between the cities with minimum distance, and she computes the total distance for each possible path, she must consider a total of 5040 paths! ▲

7. How many permutations of the letters ABCDEFGH contain the string ABC?

Solution:

Because the letters ABC must occur as a block, we can find the answer by finding the number of permutations of six objects, namely, the block ABC and the individual letters D, E, F, G, and H. Because these six objects can occur in any order, there are 6! = 720 permutations of the letters ABCDEFGH in which ABC occurs as a block. ▲ Combinations We now turn our attention to counting unordered selections of objects. We begin by solving a question posed in the introduction to this section of the chapter.

8. How many different committees of three students can be formed from a group of four students?

Solution:

To answer this question, we need only find the number of subsets with three elements from the set containing the four students. We see that there are four such subsets, one for each of the four students because choosing three students is the same as choosing one of the four students to leave out of the group. This means that there are four ways to choose the three students for the committee, where the order in which these students are chosen does not matter. ▲

Summary

The key concepts learned from this unit are: -

- We have learned what are the basics of counting.
- We have learned how to use the product rule.
- We have learned the use of the product rule in the basics of counting.
- We have learned the use of the Sum rule in the basics of counting.
- We have learned the use of the Subtraction rule in the basics of counting.
- We have learned the use of the product rule, the Sum Rule and the Subtraction rule in the basics of counting.
- We have learned the use of a tree diagram in the basics of counting.
- We have learned the Division rule in the basics of counting.
- We have learned how to find the total number of functions in the basics of counting.

Self Assessment

1. There are four bus lines between A and B and three bus lines between B and C in how many ways can a man travel by bus from A to c by way of B?
   A. 12
   B. 13
   C. 14
   D. 15

2. Suppose repetitions are not permitted. How many three- digit numbers can be formed from the six digits 2,3,5,6,7 and 9?
   A. 120
   B. 140
   C. 150
   D. 200

3. Find the number of ways that a party of seven persons can arrange themselves: in a row of seven chairs.
   A. 15!
   B. 17!
   C. 20!
   D. 22!

4. Find the number of distinct permutations that can be formed from all the letters of each word: RADAR.
   A. 30
   B. 40
   C. 50
   D. 60

5. In how many ways can four mathematics books, three history books,three chemistry books. And two sociology books be arranged on a shelf so that all books of the same subject are together?
   A. 350
   B. 450
   C. 472
   D. 879

6. Find n if p(n,2)= 72
   A. 9
   B. 8
   C. 7
   D. 5

7. In how many ways are find examinations be scheduled in a week so that no two examinations are scheduled on the same day considering Sunday as a holidays?
   A. 15
   B. 12
   C. 10
   D. 6

8. In a certain programming language, variables should be of length three and should be made up of two letters followed by a digit or of length two made up of a letter followed by a digit . How many possible variables?
   A. 250
   B. 260
   C. 720
   D. 200

9. Six boys and 6 girls are to be seated in a row, how many ways can they be seated if all boys are to be seated together and all girls are to be seated together.
   A. 6!x6!
   B. 6!x2

C. 6!x6!x2
D. 6!x6!x2x2

10. How many ways can be letter in word MISSISSIPPI can be arranged?
    A. 11!/4!4!
    B. 11!/4!2!
    C. 10!/4!4!2!
    D. 11!/4!4!2!

11. In how many ways can letters a, b, c, d, m and n be arranged if M and n always appear together
    A. 10!x3
    B. 5!x2
    C. 5!x5
    D. 10!x4

12. If repetitions are not permitted, how many four-digit number can be formed from digit 1,2,3,7,8, and 5. Total number of less than 5000.
    A. 100
    B. 180
    C. 280
    D. 380

13. A word that reads the same when read in forward or backward is called as palindrome.How many seven-letters palindromes can be formed from English alphabets?
    A. 26^2
    B. 25^4
    C. 26^4
    D. 26^5

14. Suppose repetitions are not permitted. How many of these numbers are less than 400?
    A. 30
    B. 40
    C. 50
    D. 60

15. Find the number of ways that a party of seven persons can arrange themselves: around a circular table.
    A. (n)!
    B. (n-1)!
    C. (n-2)!
    D. (n-3)!

Answer for Self Assessment

1.	A	2.	A	3.	B	4.	A	5.	C
6.	A	7.	D	8.	B	9.	C	10.	D
11.	B	12.	B	13.	C	14.	B	15.	B

Review Questions

Q1. List all the permutations of{a,b,c}.

Q2. How many permutations of {a,b,c,d,e,f,g} end with a?

Q3. Find the value of each of these quantities. a) P(6,3) b) P(6,5) c) P(8,1)

Q4. Find the number of 5-permutations of a set with nine elements.

Q5. How many possibilities are there for the win, place, and show (first, second, and third) positions in a horse race with 12 horses if all orders of finish are possible?

Q6. A group contains n men and n women. How many ways are there to arrange these people in a row if the men and women alternate?

Q7. How many permutations of the letters ABCDEFG contain a) the string BCD? b) the string CFGA? c) the strings BA and GF ? d) the strings ABC and DE? e) the strings ABC and CDE? f) the strings CBA and BED?

Q8. How many ways are there for eight men and five women to stand in a line so that no two women stand next to each other? [Hint: First position the men and then consider possible positions for the women.]

Q9. One hundred tickets numbered 1,2,3,...,100, are sold to100differentpeopleforadrawing.Fourdifferentprizes areawarded,includingagrandprize(atriptoTahiti).How many ways are there to award the prizes if a) there are no restrictions? b) the person holding ticket 47 wins the grand prize? c) the person holding ticket 47 wins one of the prizes? d) the person holding ticket 47 does not win a prize? e) the people holding tickets 19 and 47 both win prizes? f) the people holding tickets 19, 47, and 73 all win prizes? g) the people holding tickets 19, 47, 73, and 97 all win prizes? h) none of the people holding tickets 19, 47, 73, and 97 wins a prize? i) the grand prize winner is a person holding ticket 19, 47, 73, or 97? j) the people holding tickets 19 and 47 win prizes, but thepeopleholdingtickets73and97donotwinprizes?

Q10. Howmany4-permutationsofthepositiveintegersnotexceeding 100 contain three consecutive integers k, k+1, k+2, in the correct order a) where these consecutive integers can perhaps be separated by other integers in the permutation? b) where they are in consecutive positions in the permutation?

Q11. The English alphabet contains 21 consonants and five vowels. How many strings of six lowercase letters of the English alphabet contain a) exactly one vowel? b) exactly two vowels? c) at least one vowel? d) at least two vowels?

Q12. How many license plates consisting of three letters followed by three digits contain no letter or digit twice? A circular r-permutation of n people is a seating of r of these n people around a circular table, where seatingsare considered to be the same if they can be obtained from each other by rotating the table.

Q13. Find a formula for the number of circular r-permutations of n people.

Q14. How many ways are there for a horse race with three horses to finish if ties are possible? [Note: Two or three horses may tie.]

Q15. Find the value of each of these quantities. a) P(8,5) b) P(8,8) c) P(10,9).

Further Reading

- Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
- Rosen, Kenneth H., and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.
- Koshy, Thomas. *Discrete mathematics with applications*. Elsevier, 2004.
- Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 06: Counting-2

CONTENTS

CONTENTS

Objectives

Introduction

6.1 Some Revision of Basic Counting Principles:

6.2 Tree Diagrams

6.3 The Pigeonhole Principle

6.4 Combinations with Repetitions

Summary

Answer for Self Assessment

Review Questions

Further Readings

Objectives:

The Key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to

- Understand some counting related to the deck of cards
- Understand with different examples how to count the different possible ways of combination
- Understand what is a pigeonhole principle
- Understand with the different example how to count the different possible ways of combination

Introduction

Many counting problems can be solved by finding the number of ways to arrange a specified number of distinct elements of a set of a particular size, where the order of these elements matters. Many other counting problems can be solved by finding the number of ways to select a particular number of elements from a set of a particular size, where the order of the elements selected does not matter. For example, in how many ways can we select three students from a group of five students to stand in line for a picture? How many different committees of three students can be formed from a group of four students? In this section we will develop methods to answer questions such as these. Permutations: We begin by solving the first question posed in the introduction to this section, as well as related questions. This chapter develops some techniques for determining, without direct enumeration, the number of possible outcomes of a particular event or the number of elements in a set. Such sophisticated counting is sometimes called combinatorial analysis. It includes the study of combinations.

6.1 Some Revision of Basic Counting Principles:

There are two basic counting principles used throughout this chapter. The first one involves addition and the second one multiplication.

Sum Rule Principle:

Suppose some event E can occur in m ways and a second event F can occur in n ways, and suppose both

events cannot occur simultaneously. Then E or F can occur in m + n ways.

Sum Rule:

If no two events can occur at the same time, then one of the events can occur in:

n1 + n2 +n3+……. Ways

Product Rule Principle:

Suppose there is an event E which can occur in m ways and, independent of this event, there is a second eventF which can occur in n ways. Then combinations of E and F can occur in mn ways.

The above principles can be extended to three or more events. That is, suppose an event E1 can occur in n1ways, a second event E2 can occur in n2 ways, and, following E2; a third event E3 can occur in n3 ways, and soon. Then:

Product Rule:

If the events occur one after the other, then all the events can occur in the order indicated in:

n1 ·n2 ·n3 ·... ways.

Example 1

Suppose a college has 3 different history courses, 4 different literature courses, and 2 different sociology courses.

(a) The number m of ways a student can choose one of each kind of courses is:

m = 3(4)(2) = 24

(b) The number n of ways a student can choose just one of the courses is:

n = 3 + 4 + 2 = 9

There is a set-theoretical interpretation of the above two principles. Specifically, suppose n(A) denotes the number of elements in a set A. Then:

(1) Sum Rule Principle: Suppose A and B are disjoint sets. Then

n(A ∪ B) = n(A) + n(B)

(2) Product Rule Principle: Let A × B be the Cartesian product of sets A and B. Then

n(A × B) = n(A) ·n(B)

MATHEMATICAL FUNCTIONS:

We discuss two important mathematical functions frequently used in combinatory.

Factorial Function

The product of the positive integers from 1 to n inclusive is denoted by n!, read "n factorial." Namely:

n! = 1 ·2 ·3 ·... ·(n−2)(n−1)n = n(n−1)(n−2) ·... ·3 ·2 ·1

Accordingly, 1! = 1 and n! = n(n − l)!. It is also convenient to define 0! = 1

Example 2

3 != 3. 2. 1= 6, 4! = 4 . 3. 2.1 = 24, 5 = 5. 4. ! = 5(24) = 120

$\frac{12.11.10}{3.2.1} = \frac{12.11.10.9!}{3.2.1.9.1} = \frac{12!}{319!}$ and, more generally

$$\frac{n(n-1)\cdots(n-r+1)}{r(r-1)\cdots 3\cdot 2\cdot 1} = \frac{n(n-1)\cdots(n-r+1)(n-r)!}{r(r-1)\cdots 3\cdot 2\cdot 1\cdot(n-r)!} = \frac{n!}{r!(n-r)!}$$

For large n , one uses sterling 's approximation (where e = 2.7128…..)

$n! = \sqrt{2\pi n} n^n e^{-n}$

Binomial Coefficients:

The symbol $\binom{n}{r}$, read "nCr" or "n Choose r," where r and n are positive integers with $r \leq n$, is defined as follows:

$\binom{n}{r} = \frac{n(n-1)\cdots(n-r+1)\,r(r-1)}{r(r-1)\ldots 3\cdot 2\cdot 1}$ or equivalently $\binom{n}{r} = \frac{n!}{r!(n-r)!}$

Note that n − (n − r) = r. This yields the following important relation

Lemma: $\binom{n}{n-r} = \binom{n}{r}$ or equivalently $\binom{n}{a} = \binom{n}{b}$ Where a + b = n

Motivated by the fact that we defined 0! = 1, we define

$\binom{n}{0} = \frac{n!}{0!n!} = 1$ and $\binom{0}{0} = \frac{0!}{0!0!} = 1$

Example3

$\binom{8}{2} = \frac{8\cdot 7}{2\cdot 1} = 28$; $\binom{9}{4} = \frac{9\cdot 8\cdot 7\cdot 6}{4\cdot 3\cdot 2\cdot 1} = 126$; $\binom{12}{5} = \frac{12\cdot 11\cdot 10\cdot 9\cdot 8}{5\cdot 4\cdot 3\cdot 2\cdot 1} = 792$

Note that $\binom{n}{r}$ has exactly r factors in both the numerator and the denominator.

Suppose we want to compute $\binom{10}{7}$. There will be 7 factors in both the numerator and the denominator.

However, 10 − 7 = 3. Thus, we use Lemma 5.1 to compute:

$\binom{10}{7} = \binom{10}{3} = \frac{10\cdot 9\cdot 8}{3\cdot 2\cdot 1} = 120.$

6.2 Tree Diagrams

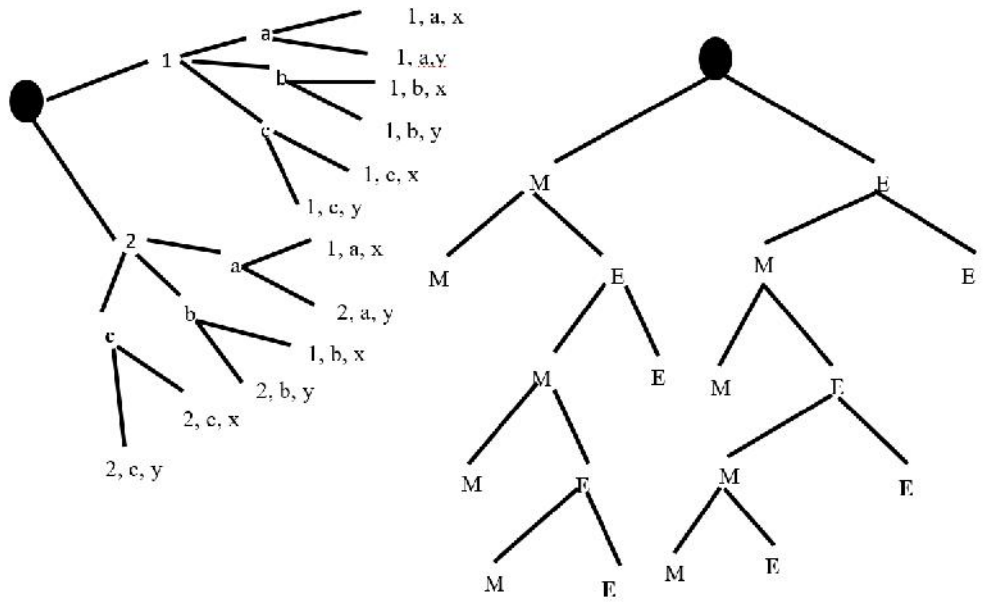
A tree diagram is a device used to enumerate all the possible outcomes of a sequence of events where each event can occur in a finite number of ways. The construction of tree diagrams is illustrated in the following example.

Example4

4 Find the product set A × B × C, where A = {l, 2}, B = {a, b, c}, C = {x, y}.

The tree diagram for A × B × C appears in Fig. 5-2(a). Here the tree is constructed from left to right, and the number of branches at each point corresponds to the possible outcomes of the next event. Each endpoint (leaf) of the tree is labelled by the corresponding element of A × B × C. As noted previously, A × B × C has n = 2(3)(2) = 12 elements

TECHNIQUES OF COUNTING:



Mark and Erik are to play a tennis tournament. The first person to win two games in a row or who wins a total of three games wins the tournament. Find the number of ways the tournament can occur.

The tree diagram showing the possible outcomes of the tournament appears in figure Here the tree is constructed from top-down rather than from left-right. (That is, the "root" is on the top of the tree.) Note that there are 10 endpoints, and the endpoints correspond to the following 10 ways the tournament can occur:

MM, MEMM, MEMEM, MEMEE, MEE, EMM, EMEMM, EMEME, EMEE, EE

The path from the beginning (top) of the tree to the endpoint describes who won which game in the tournament

Binomial Coefficients and Pascal's Triangle:

The numbers (n/r) are called binomial coefficients since they appear as the coefficients in the expansion of

$(a + b)^n$. Specifically

Theorem (Binomial Theorem) : $(a + b)^n = \sum_{k=0}^{n} \binom{n}{r} a^{n-k} - b^k$

The coefficients of the successive powers of a + b can be arranged in a triangular array of numbers, called

Pascal's triangle, as pictured in Fig. 5-1. The numbers in Pascal's triangle have the following interesting properties:

(i) The first and last number in each row is 1.

(ii) Every other number can be obtained by adding the two numbers appearing above it. For example:

10 = 4 + 6, 15 = 5 + 10, 20 = 10 + 10.

Since these numbers are binomial coefficients, we state the above property formally.

$$(a+ b)^0 = 1$$

$(a+b)^1 = a + b$

$(a+b)^2 = a^2 + b^2 + 2ab$

$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

$(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$

$(a+b)^5 = a^5 + 5a^4b + 10a^3b2 + 10a^2b^3 + 5ab^4 + b^5$

$(a+b)^6 = a + 6a^5b + 15a^4b^2 + 20a^3b^3 + 15a^2b^4 + 6ab^5 + b^6$

PASCAL' S TRIANGLE

1

1 1

1 2 1

1 3 3 3

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 6 15 20 15 6 1

Theorem : $\binom{n+1}{r} = \binom{n}{r-1} + \binom{n}{r}$

Example 5 In how many ways can we select three students from a group of five students to stand in line for a picture? In how many ways can we arrange all five of these students in a line for a picture? Solution: First, note that the order in which we select the student's matters. There are five ways to select the first student to stand at the start of the line. Once this student has been selected, there are four ways to select the second student in the line. After the first and second students have been selected, there are three ways to select the third student in the line. By the product rule, there are $5 \cdot 4 \cdot 3 = 60$ ways to select three students from a group of five students to stand in line for a picture. To arrange all five students in a line for a picture, we select the first student in five ways, the second in four ways, the third in three ways, the fourth in two ways, and the fifth in one way. Consequently, there are $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$ ways to arrange all five students in a line for a picture. ▲

Example 6 6 Let S = {1, 2, 3}. The ordered arrangement 3, 1, 2 is a permutation of S. The ordered arrangement 3, 2 is a 2-permutation of S. ▲ The number of r-permutations of a set with n elements is denoted by P (n, r). We can find P (n, r) using the product rule.

Example 7 Let S = {a, b, c}. The 2-permutations of S are the ordered arrangements a, b; a, c; b, a; b, c; c, a; and c, b. Consequently, there are six 2-permutations of this set with three elements. There are always six 2-permutations of a set with three elements. There are three ways to choose the first element of the arrangement. There are two ways to choose the second element of the arrangement because it must be different from the first element. Hence, by the product rule, we see that P (3, 2) = $3 \cdot 2 = 6$. the first element. By the product rule, it follows that P (3, 2) = $3 \cdot 2 = 6$. ▲ We now use the product rule to find a formula for P (n, r) whenever n and r are positive integers with $1 \le r \le n$. THEOREM: If n is a positive integer

and r is an integer with $1 \le r \le n$, then there are $P(n, r) = n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of a set with n distinct elements. Proof: We will use the product rule to prove that this formula is correct. The first element of the permutation can be chosen in n ways because there are n elements in the set. There is $n - 1$ way to choose the second element of the permutation because there are $n - 1$ elements left in the set after using the element picked for the first position. Similarly, there are $n - 2$ ways to choose the third element, and so on, until there are exactly $n - (r - 1) = n - r + 1$ ways to choose the rth element. Consequently, by the product rule, there are $n(n-1)(n-2)\cdots(n-r+1)$ r-permutations of the set. Note that $P(n, 0) = 1$ whenever n is a non-negative integer because there is exactly one way to order zero elements. That is, there is exactly one list with no elements in it, namely the empty list. We now state a useful corollary of Theorem 1. COROLLARY 1

If n and r are integers with $0 \le r \le n$, then $P(n, r) = \frac{n!}{(n-r)!}$. Proof: When n and r are integers with $1 \le r \le n$, by Theorem 1 we have $P(n, r) = n(n-1)(n-2)\cdots = \frac{n!}{(n-r)!}$. Because $\frac{n!}{(n-0)!} = \frac{n!}{n!} = 1$ whenever n is a non-negative integer, we see that the formula $P(n, r) = \frac{n!}{(n-r)!}$ also holds when $r = 0$.

By Theorem 1 we know that if n is a positive integer, then $P(n, n) = n!$. We will illustrate this result with some examples.

Example 8 How many ways are there to select a first-prize winner, a second-prize winner, and a third-prize winner from 100 different people who have entered a contest? Solution: Because it matters which person wins which prize, the number of ways to pick the three prize winners is the number of ordered selections of three elements from a set of 100 elements, that is, the number of 3-permutations of a set of 100 elements. Consequently, the answer is $P(100, 3) = 100 \cdot 99 \cdot 98 = 970{,}200$. ▲

Example 9 Suppose that there are eight runners in a race. The winner receives a gold medal, the second-place finisher receives a silver medal, and the third-place finisher receives a bronze medal. How many different ways are there to award these medals, if all possible outcomes of the race can occur and there are no ties? Solution: The number of different ways to award the medals is the number of 3-permutations of a set with eight elements. Hence, there are $P(8, 3) = 8 \cdot 7 \cdot 6 = 336$ possible ways to award the medals. ▲

Example 10 Suppose that a saleswoman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities? Solution: The number of possible paths between the cities is the number of permutations of seven elements because the first city is determined, but the remaining seven can be ordered arbitrarily. Consequently, there are $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$ ways for the saleswoman to choose her tour. If, for instance, the saleswoman wishes to find the path between the cities with minimum distance, and she computes the total distance for each possible path, she must consider a total of 5040 paths! ▲

*Example*11

How many permutations of the letters ABCDEFGH contain the string ABC? Solution: Because the letters ABC must occur as a block, we can find the answer by finding the number of permutations of six objects, namely, the block ABC and the individual letters D, E, F, G, and H. Because these six objects can occur in any order, there are 6! = 720 permutations of the letters ABCDEFGH in which ABC occurs as a block. ▲ Combinations We now turn our attention to counting unordered selections of objects. We begin by solving a question posed in the introduction to this section of the chapter. EXAMPLE 8 How many different committees of three students can be formed from a group of four students? Solution: To answer this question, we need only find the number of subsets with three elements from the set containing the four students. We see that there are four such subsets, one for each of the four students because choosing three students is the same as choosing one of the four students to leave out of the group. This means that there are four ways to choose the three students for the committee, where the order in which these students are chosen does not matter. ▲

*Example*12

Illustrates that many counting problems can be solved by finding the number of subsets of a particular size of a set with n elements, where n is a positive integer. An r-combination of elements of a set is an unordered selection of r elements from the set. Thus, an r-combination is simply a subset of the set with r elements.

*Example*13

Let S be the set {1, 2, 3, 4}. Then {1, 3, 4} is a 3-combination from S. (Note that {4, 1, 3} is the same 3-combination as {1, 3, 4}, because the order in which the elements of a set are listed does not matter.) ▲ The number of r-combinations of a set with n distinct elements is denoted by C(n, r). Note that C(n, r) is also denoted by n r and is called a binomial coefficient. We will learn where this terminology comes from in Section 6.4.

*Example*14

We see that C(4, 2) = 6, because the 2-combinations of {a, b, c, d} are the six subsets {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, and {c, d}. ▲ We can determine the number of r-combinations of a set with n elements using the formula for the number of r-permutations of a set. To do this, note that the r-permutations of a set can be obtained by first forming r-combinations and then ordering the elements in these combinations. The proof of Theorem 2, which gives the value of C(n, r), is based on this observation.

Let S be a set with n elements. A combination of these n elements taken r at a time is any selection of r ofthe elements where order does not count. Such a selection is called an r-combination; it is simply a subset of Swith r elements. The number of such combinations will be denoted byC(n, r) (other texts may use nCr, Cn,r, or Cnr. Before we give the general formula for C(n, r), we consider a special case

*Example*15

Find the number of combinations of 4 objects, A, B, C, D, taken 3 at a time.

Each combination of three objects determines 3! = 6 permutations of the objects as follows:

ABC: ABC, ACB, BAC, BCA, CAB, CBA

ABD: ABD, ADB, BAD, BDA, DAB, DBA

ACD: ACD, ADC, CAD, CDA, DAC, DCA

BCD: BDC, BDC, CBD, CDB, DBC, DCB

Thus the number of combinations multiplied by 3! gives us the number of permutations; that is,

C(4, 3) ·3! = P (4, 3) or C(4, 3) = (P (4,3))/3!

But P (4, 3) = 4 ·3 ·2 = 24 and 3! = 6; hence C(4, 3) = 4 as noted above.

As indicated above, any combination of n objects taken r at a time determines r! permutations of the objects in the combination; that is,

P (n, r) = r! C(n, r)

THEOREM: The number of r-combinations of a set with n elements, where n is a nonnegative integer and r is an integer with $0 \le r \le n$, equals

C(n, r) =. $\frac{n!}{r!(n-r)!}$

Proof: The P (n, r) r-permutations of the set can be obtained by forming the C(n, r) r-combinations of the set, and then ordering the elements in each r-combination, which can be done in P (r, r) ways. Consequently, by the product rule, P (n, r) = C(n, r) ·P (r, r). This is C(n, r) = $\frac{P(n,r)}{P(r,r)} = \frac{n!/(n-r)!}{r!/(r-r)!} = \frac{n!}{r!(n-r)!}$

We can also use the division rule for counting to construct a proof of this theorem. Because the order of elements in a combination does not matter and there are P (r, r) ways to order elements in an r-combination of n elements, each of the C(n, r) r-combinations of a set with n elements corresponds to exactly P (r, r) r-permutations. Hence, by the division rule, C(n, r) = $\frac{P(n,r)}{P(r,r)}$ which implies as before that C(n, r) = $\frac{n!}{r!(n-r)!}$.

OR

C(n, r) =! (P (n,r))/r! = n!/(r!(n - r)!)

Recall that the binomial coefficient (n/r) was defined to be n (n!)/(r!(n - r)!)!; hence C(r, n) =(n/r) We shall use C(n, r) and (n/r) interchangeably

*Example*16 A farmer buys 3 cows, 2 pigs, and 4 hens from a man who has 6 cows, 5 pigs, and 8 hens. Findthe number m of choices that the farmer has.

The farmer can choose the cows in C(6, 3) ways, the pigs in C(5, 2) ways, and the hens in C(8, 4) ways.

Thus the number m of choices follows:

m = (6/3)(5/2)(8/4)=6.5.4/3.2.1 5.4/2.1 8.7.6.5/4.3.2.1=20.10.70=14000

The formula in Theorem 2, although explicit, is not helpful when C(n, r) is computed for large values of n and r. The reasons are that it is practical to compute exact values of factorials exactly only for small integer values, and when floating-point arithmetic is used, the formula in Theorem 2 may produce a value that is not an integer. When computing C(n, r), first note that when we cancel out (n − r)! from the numerator and denominator of the expression for C(n, r) in Theorem 2, we obtain C(n, r) = $\frac{n!}{r!(n-r)!}$. = $\frac{n(n-1)\cdots(n-r+1)}{r!}$

Consequently, to compute C(n, r) you can cancel out all the terms in the larger factorial in the denominator from the numerator and denominator, then multiply all the terms that do not cancel in the numerator and finally divide by the smaller factorial in the denominator. [When doing this calculation by hand, instead of by machine, it is also worthwhile to factor out common

factors in the numerator $n(n-1)\cdots(n-r+1)$ and in the denominator $r!$.] Note that many calculators have a built-in function for C(n, r) that can be used for relatively small values of n and r and many computational programs can be used to find C(n, r). [Such functions may be called choose(n, k) or binom(n, k)]. Example illustrates how C(n, k) is computed when k is relatively small compared to n and when k is close to n. It also illustrates a key identity enjoyed by the numbers C(n, k).

*Example*17 How many poker hands of five cards can be dealt from a standard deck of 52 cards? Also, how many ways are there to select 47 cards from a standard deck of 52 cards? Solution: Because the order in which the five cards are dealt from a deck of 52 cards does not matter, there are.

$$C(52, 5) = \frac{52!}{5!47!}$$

different hands of five cards that can be dealt. To compute the value of C(52, 5), first divide the numerator and denominator by 47! to obtain $C(52, 5) = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}$

This expression can be simplified by first dividing the factor 5 in the denominator into the factor 50 in the numerator to obtain a factor 10 in the numerator, then dividing the factor 4 in the denominator into the factor 48 in the numerator to obtain a factor of 12 in the numerator, then dividing the factor 3 in the denominator into the factor 51 in the numerator to obtain a factor of 17 in the numerator, and finally, dividing the factor 2 in the denominator into the factor 52 in the numerator to obtain a factor of 26 in the numerator. We find that $C(52, 5) = 26 \cdot 17 \cdot 10 \cdot 49 \cdot 12 = 2{,}598{,}960$. Consequently, there are 2,598,960 different poker hands of five cards that can be dealt from a standard deck of 52 cards. Note that there are

$$C(52, 5) = \frac{52!}{5!47!}$$

different ways to select 47 cards from a standard deck of 52 cards. We do not need to compute this value becauseofC(52, 47) = C(52, 5). (Only the order of the factors 5! and 47! is different in the denominators in the formulae for these quantities.) It follows that there are also 2,598,960 different ways to select 47 cards from a standard deck of 52 cards. ▲

In Example 11 we observed that C(52, 5) = C(52, 47). This is a special case of the useful identity for the number of r-combinations of a set given in Corollary 2. COROLLARY 2 Let n and r be nonnegative integers with $r \le n$. Then C(n, r) = C(n, n − r). Proof: From Theorem 2 it follows that $C(n, r) = \frac{n!}{r!(n-r)!}$ and $C(n, n - r) = \frac{n!}{(n-r)![n-(n-r)]!} = \frac{n!}{(n-r)!r!}$

Hence, C(n, r) = C(n, n − r). We can also prove Corollary 2 without relying on algebraic manipulation. Instead, we can use combinatorial proof. We describe this important type of proof in Definition 1. DEFINITION: A combinatorial proof of an identity is a proof that uses counting arguments to prove that both sides of the identity count the same objects but in different ways or a proof that is based on showing that there is a bijection between the sets of objects counted by the two sides of the identity. These two types of proofs are called double counting proofs and bijective proofs, respectively Many identities involving binomial coefficients can be proved using combinatorial proofs. We now show how to prove Corollary 2 using a combinatorial proof. We will provide both a double counting proof and a bijective proof, both based on the same basic idea

Proof: We will use a bijective proof to show that C(n, r) = C(n, n − r) for all integers n and r with $0 \le r \le n$. Suppose that S is a set with n elements. The function that maps a subset A of S to A is a bijection between subsets of S with r elements and subsets with n − r elements (as the reader should verify). The

identity C(n, r) = C(n, n − r) follows because when there is a bijection between two finite sets, the two sets must have the same number of elements. Alternatively, we can reformulate this argument as a double counting proof. By definition, the number of subsets of S with r elements equals C(n, r). But each subset A of S is also determined by specifying which elements are not in A, and so are in A. Because the complement of a subset of S with r elements has n − r elements, there are also C(n, n − r) subsets of S with r elements. It follows that C(n, r) = C(n, n − r).

Example 18 How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school? Solution: The answer is given by the number of 5-combinations of a set with 10 elements. By Theorem 2, the number of such combinations is $C(10, 5) = \frac{10!}{5!\,5!} = 252$.

Example 19 A group of 30 people have been trained as astronauts to go on the first mission to Mars. How many ways are there to select a crew of six people to go on this mission (assuming that all crew members have the same job)? Solution: The number of ways to select a crew of six from the pool of 30 people is the number of 6-combinations of a set with 30 elements, because the order in which these people are chosen does not matter. By Theorem 2, the number of such combinations is

$$C(30,6) = \frac{30!}{6!\,24!} = \frac{30 \cdot 29 \cdot 28 \cdot 27 \cdot 26 \cdot 25.}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 593{,}775$$

Example 20 Suppose that there are 9 faculty members in the mathematics department and 11 in the computer science department. How many ways are there to select a committee to develop a discrete mathematics course at a school if the committee is to consist of three faculty members from the mathematics department and four from the computer science department? Solution: By the product rule, the answer is the product of the number of 3-combinations of a set with nine elements and the number of 4-combinations of a set with 11 elements. By Theorem 2, the number of ways to select the committee is

$C(9, 3) \cdot C(11, 4) = \frac{9!}{3!6!} \cdot \frac{11!}{4!7!} = 84 \cdot 330 = 27{,}720.$

6.3 The Pigeonhole Principle

Introduction

Many results in combinational theory come from the following almost obvious statement.

Pigeonhole Principle: If n pigeonholes are occupied by n + 1 or more pigeons, then at least one pigeonhole is occupied by more than one pigeon.

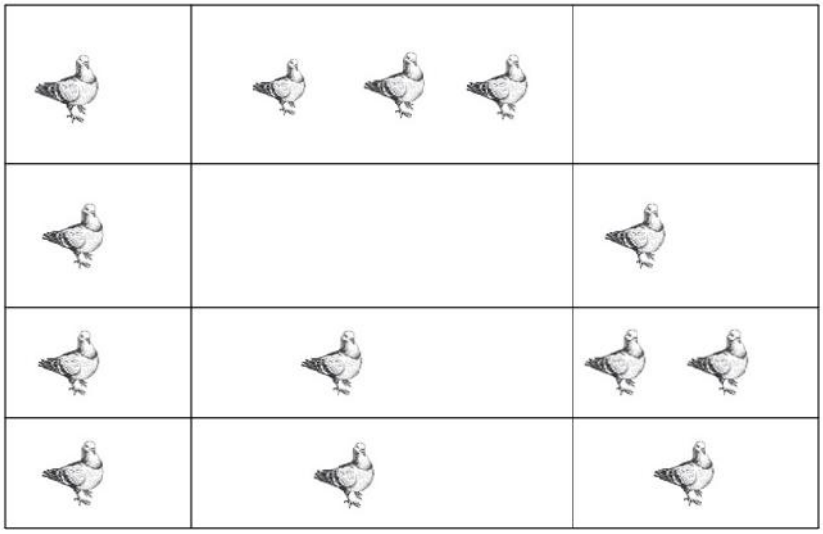
This principle can be applied to many problems where we want to show that a given situation can occur.

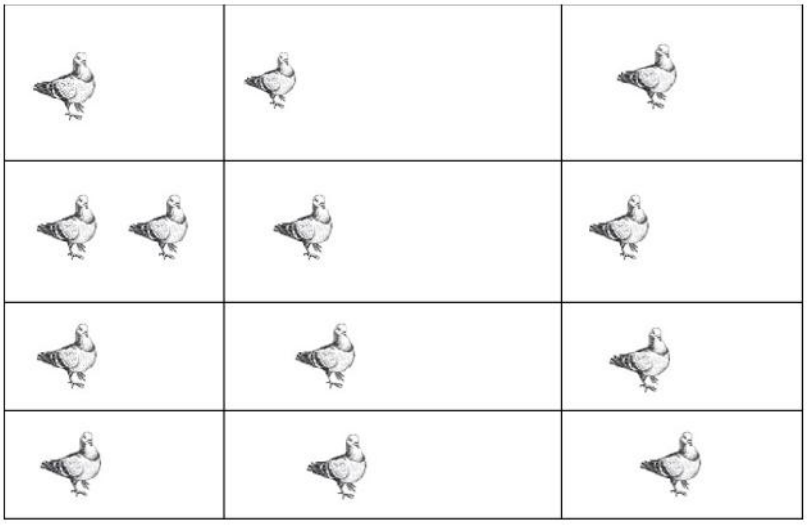
Suppose that a flock of 20 pigeons flies into a set of 19 pigeonholes to roost. Because there are 20 pigeons but only 19 pigeonholes, a least one of these 19 pigeonholes must have at least two pigeons in it. To see why this is true, note that if each pigeonhole had at most one pigeon in it, at most 19 pigeons, one per hole, could be accommodated. This illustrates a general principle called the pigeonhole principle, which states that if there are more pigeons than pigeonholes, then there must be at least one pigeonhole with at least two pigeons in it (see Figure 1). Of course, this principle applies to other objects besides pigeons and pigeonholes.

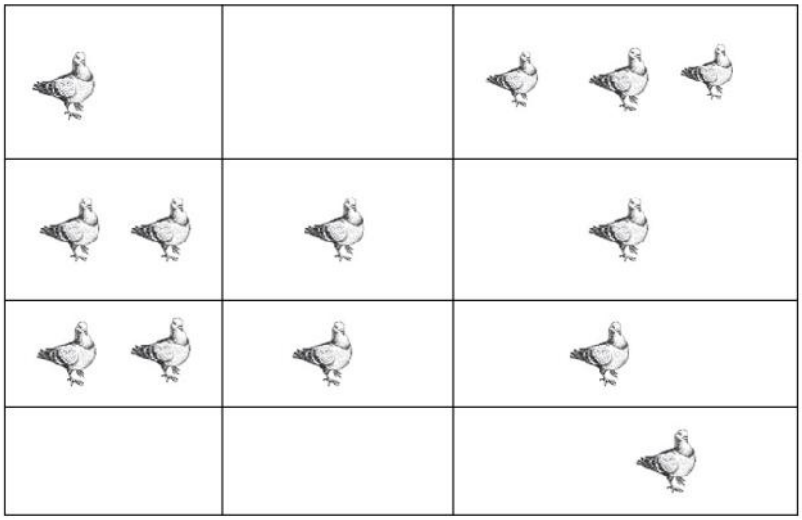
THEOREM:

THE PIGEONHOLE PRINCIPLE

If k is a positive integer and k + 1 or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.







Proof: We prove the pigeonhole principle using proof by contraposition. Suppose that none of the k boxes contains more than one object. Then the total number of objects would be at most k. This is a contradiction because there are at least k + 1 objects. The pigeonhole principle is also called the Dirichlet drawer principle, after the nineteenth-century German mathematician G. Lejeune Dirichlet, who often used this principle in his work. (Dirichlet was not the first person to use this principle; a demonstration that there were at least two Parisians with the same number of hairs on their heads dates back to the 17th century— see Exercise 33.) It is an important additional proof technique supplementing those we have developed in earlier chapters. We introduce it in this chapter because of its many important applications to combinatorics. We will illustrate the usefulness of the pigeonhole principle. We first show that it can be used to prove a useful corollary about functions.

COROLLARY 1 A function f from a set with k + 1 or more elements to a set with k elements is not one-to-one. Proof: Suppose that for each element y in the co domainof f we have a box that contains all elements x of the domain of f such that f (x) = y. Because the domain contains k + 1 or more elements and the co domain contains only k elements, the pigeonhole principle tells us that one of these boxes contains two or more elements x of the domain. This means that f cannot be one-to-one. Examples 1–3 show how the pigeonhole principle is used.

Example 21 Suppose a department contains 13 professors, then two of the professors (pigeons) were born in the same month (pigeonholes).

(b) Find the minimum number of elements that one needs to take from the set S = {1, 2, 3,..., 9} to be sure that two of the numbers add up to 10.

Here the pigeonholes are the five sets {1, 9}, {2, 8}, {3, 7}, {4, 6}, {5}. Thus any choice of six elements

(pigeons) of S will guarantee that two of the numbers add up to ten.

The Pigeonhole Principle is generalized as follows.

Example 22 Among any group of 367 people, there must be at least two with the same birthday, because there are only 366 possible birthdays. ▲

Example 23 In any group of 27 English words, there must be at least two that begin with the same letter, because there are 26 letters in the English alphabet. ▲

Example 24 How many students must be in a class to guarantee that at least two students receive the same score on the final exam, if the exam is graded on a scale from 0 to 100 points? Solution: There are 101 possible scores on the final. The pigeonhole principle shows that among any 102 students there must be at least 2 students with the same score. ▲ G. LEJEUNE DIRICHLET (1805–1859) G. Lejeune Dirichlet was born into a Belgian family living near Cologne, Germany. His father was a postmaster. He became passionate about mathematics at a young age. He was spending all his spare money on mathematics books by the time he entered secondary school in Bonn at the age of 12.At 14 he entered the Jesuit College in Cologne, and at 16 he began his studies at the University of Paris. In 1825 he returned to Germany and was appointed to a position at the University of Breslau. In 1828 he moved to the University of Berlin. In 1855 he was chosen to succeed Gauss at the University of Göttingen. Dirichlet is said to be the first person to master Gauss's DisquisitionesArithmeticae, which appeared 20 years earlier. He is said to have kept a copy at his side even when he travelled. Dirichlet made many important discoveries in number theory, including the theorem that there are infinitely many primes in arithmetical progressions an + b when a and b are relatively prime. He proved the n = 5 case of Fermat's last theorem, that there are no nontrivial solutions in integers to x5 + y5 = z5. Dirichlet also made many contributions to analysis. Dirichlet was considered to be an excellent teacher who could explain ideas with great clarity. He was married to Rebecca Mendelssohn, one of the sisters of the composer Frederick Mendelssohn.

The pigeonhole principle is a useful tool in many proofs, including proofs of surprising results, such as that given in Example 4.

Example 25 Show that for every integer n there is a multiple of n that has only 0s and 1s in its decimal expansion. Solution: Let n be a positive integer. Consider the n + 1 integers 1, 11, 111,..., 11 ... 1 (where the last integer in this list is the integer with n + 1 1s in its decimal expansion). Note that there are n possible remainders when an integer is divided by n. Because there are n + 1 integers in this list, by the pigeonhole principle there must be two with the same remainder when divided by n. The larger of these integers less the smaller one is a multiple of n, which has a decimal expansion consisting entirely of 0s and 1s.

The Generalized Pigeonhole Principle The pigeonhole principle states that there must be at least two objects in the same box when there are more objects than boxes. However, even more can be said when the number of objects exceeds a multiple of the number of boxes. For instance, among any set of 21 decimal digits there must be 3 that are the same. This follows because when 21 objects are distributed into 10 boxes, one box must have more than 2 objects.

Generalized Pigeonhole Principle: If n pigeonholes are occupied by kn + 1 or more pigeons, where k is a positive integer, then at least one pigeonhole is occupied by k + 1 or more pigeons.

OR

THEOREM 2

THE GENERALIZED PIGEONHOLE PRINCIPLE

If N objects are placed into k boxes, then there is at least one box containing at least N/k objects. Proof: We will use a proof by contraposition. Suppose that none of the boxes contains more than N/k− 1 objects. Then, the total number of objects is at most

$$k\left(\frac{N}{K}-1\right) < K\left(\frac{N,}{K}+1\right)-1 = N$$

where the inequality N/k < (N/k) + 1 has been used. This is a contradiction because there are a total of N objects. A common type of problem asks for the minimum number of objects such that at least r of these objects must be in one of k boxes when these objects are distributed among the boxes. When we have N objects, the generalized pigeonhole principle tells us there must be at least r objects in one of the boxes as long as N/k ≥ r. The smallest integer N with N/k > r − 1, namely, N = k(r − 1) + 1, is the smallest integer satisfying the inequality N/k ≥ r. Could a smaller value of N

suffice? The answer is no, because if we had k(r − 1) objects, we could put r − 1 of them in each of the k boxes and no box would have at least r objects. When thinking about problems of this type, it is useful to consider how you can avoid having at least r objects in one of the boxes as you add successive objects. To avoid adding a rth object to any box, you eventually end up with r − 1 objects in each box. There is no way to add the next object without putting anrth object in that box. Examples 5–8 illustrate how the generalized pigeonhole principle is applied

Example 25 Find the minimum number of students in a class to be sure that three of them are born in the same month. Here the n = 12 months are the pigeonholes, and k + 1 = 3 so k = 2. Hence among any kn + 1 = 25 students (pigeons), three of them are born in the same month.

Example 26 Among 100 people there are at least 100/12 = 9 who were born in the same month. ▲

Example 27 What is the minimum number of students required in a discrete mathematics class to be sure that at least six will receive the same grade, if there are five possible grades, A, B, C, D, and F? Solution: The minimum number of students needed to ensure that at least six students receive the same grade is the smallest integer N such that N/5 = 6. The smallest such integer is N = 5 · 5 + 1 = 26. If you have only 25 students, it is possible for there to be five who have received each grade so that no six students have received the same grade. Thus ,26 is the minimum number of students needed to ensure that least six students will receive the same grade. ▲

Example 28 How many cards must be selected from a standard deck of 52 cards to guarantee that at least three cards of the same suit are chosen? A standard deck of 52 cards has 13 kinds of cards, with four cards of each of kind, one in each of the four suits, hearts, diamonds, spades, and clubs. b) How many must be selected to guarantee that at least three hearts are selected?

Solution: a) Suppose there are four boxes, one for each suit, and as cards are selected they are placed in the box reserved for cards of that suit. Using the generalized pigeonhole principle, we see that if N cards are selected, there is at least one box containing at least N/4 cards. Consequently, we know that at least three cards of one suit are selected if N/4 ≥ 3. The smallest integer N such that N/4 ≥ 3 is N = 2 · 4 + 1 = 9, so nine cards suffice. Note that if eight cards are selected, it is possible to have two cards of each suit, so more than eight cards are needed. Consequently, nine cards must be selected to guarantee that at least three cards of one suit are chosen. One good way to think about this is to note that after the eighth card is chosen, there is no way to avoid having a third card of some suit. b) We do not use the generalized pigeonhole principle to answer this question, because we want to make sure that there are three hearts, not just three cards of one suit. Note that in the worst case, we can select all the clubs, diamonds, and spades, 39 cards in all, before we select a single heart. The next three cards will be all hearts, so we may need to select42 cards to get three hearts

Example 29 What is the least number of area codes needed to guarantee that the 25 million phones in a state can be assigned distinct 10-digit telephone numbers? (Assume that telephone numbers are of the form NXX-NXX-XXXX, where the first three digits form the area code, N represents a digit from 2 to 9 inclusive, and X represents any digit.) Solution: There are eight million different phone numbers

of the form NXX-XXXX (as shown in Example 8 of Section 6.1). Hence, by the generalized pigeonhole principle, among 25 million telephones, at least 25,000,000/8,000,000= 4 of them must have identical phone numbers. Hence, at least four area codes are required to ensure that all 10-digit numbers are different.

Example 30 although not an application of the generalized pigeonhole principle, makes use of similar principles.

Example 31 Suppose that a computer science laboratory has 15 workstations and 10 servers. A cable can be used to directly connect a workstation to a server. For each server, only one direct connection to that server can be active at any time. We want to guarantee that at any time any set of 10 or fewer workstations can simultaneously access different servers via direct connections. Although we could do this by connecting every workstation directly to every server (using 150 connections), what is the minimum number of direct connections needed to achieve this goal? Solution: Suppose that we label the workstations W1, W2,...,W15 and the servers S1, S2,...,S10. Furthermore, suppose that we connect Wk to Sk for k = 1, 2,..., 10 and each of W11, W12, W13, W14, and W15 to all 10 servers. We have a total of 60 direct connections. Clearly, any set of 10 or fewer workstations can simultaneously access different servers. We see this by noting that if workstation Wj is included with $1 \le j \le 10$, it can access server Sj, and for each workstation Wk with $k \ge 11$ included, there must be a corresponding workstation w_j with $1 \le j \le 10$ not included, so Wk can access server Sj. (This follows because there are at least as many available servers Sj as there are workstations Wj with $1 \le j \le 10$ not included.) Now suppose there are fewer than 60 direct connections between workstations and servers. Then some server would be connected to at most 59/10 = 5 workstations. (If all servers were connected to at least six workstations, there would be at least $6 \cdot 10 = 60$ direct connections.) This means that the remaining nine servers are not enough to allow the other 10 workstations to simultaneously access different servers. Consequently, at least 60 direct connections are needed. It follows that 60 is the answer.

6.4 Combinations with Repetitions

Consider the following problem. A bakery makes only M = 4 kinds of cookies: apple (a), banana (b),

carrot (c), dates (d). Find the number of ways a person can buy r = 8 of the cookies.

Observe that order does not count. This is an example of combinations with repetitions. In particular, each combination can be listed with the a's first, then the b's, then the c's, and finally the d's. Four such combinations follow:

r1= aa, bb, cc, dd; r2 = aaa, c, ddd; r3 = bbbb, c, ddd; r4 = aaaaa, ddd.

Suppose we want to code the above combinations using only two symbols, say 0 and 1. This can be done by letting 0 denote a cookie, and letting 1 denote a change from one kind of cookie to another. Then each combination will require r = 8 zeros, one for each cookie, and M − 1 = 3 ones, where the first one denotes the change from a to b, the second one from b to c, and the third one from c to d. Thus the above four combinations will be coded as follows:

r1 = 00100100100, r2 = 00001101000, r3 = 10000101000, r4 = 00000111000.

Counting the number m of these "codewords" is easy. Each codeword contains R + M − 1 = 11 digits where r = 8 are 0's and hence M − 1 = 3 are 1's. Accordingly,

M = C(11, 8) = C(11, 3) = 11.10.9/3.2.1 = 165

A similar argument gives us the following theorem.

Suppose there are M kinds of objects. Then the number of combinations of r such objects is

C(r + M – 1, r) = C(r + M − 1, M − 1).

Summary

The key concepts learned from this unit are: -

- We have learned how to count the different possible ways of combination.
- We have learnedsome counting related to the deck of cards with different examples
- We have learned what a pigeonhole principle is.
- We have learned with different examples how to count the different possible ways of combination.

Self Assessment

Q1.In how many ways can a committee consisting of three men and two women be chosen from seven men and five women?

A. 350
B. 450
C. 500
D. 600

Q2. A bag contains six white marbles and five red marbles.Find the number of ways four marbles can be drawn from the bag if they can be any color

Ans.

A. 330
B. 340
C. 350
D. 700

Q3. How many committees of five with a given chairperson can be selected from 12 persons?

A. 3960

B. 3800

C. 4000

D. 7000

Q4. Out of 12 employees, a group of four trainees is to be sent for software testing and QA trainingfor one month.

In how many ways can the four employees be selected?

A. 490
B. 495
C. 466
D. 120

Q5. There are 50 students in each of the senior and junior classes. Each class has 25 male and 25 female students. In how many ways can an eight student committee be formed so that there are four female and three juniors in the committee?

A. $(25_{C_3}+25_{C_1}+25_{C_4})+2\ (25_{C_3}+25_{C_1}+25_{C_2}+25_{C_2})$
B. $2\ (25_{C_3}+25_{C_1}+25_{C_4})+(25_{C_3}+25_{C_1}+25_{C_2}+25_{C_2})$
C. $2\ (25_{C_3}+25_{C_1}+25_{C_4})+2*(25_{C_3}+25_{C_1}+25_{C_2}+25_{C_2})$
D. $(25_{C_3}+25_{C_1}+25_{C_4})+(25_{C_3}+25_{C_1}+25_{C_2}+25_{C_2})$

Q6. There are 12 students in a class.in how many ways can the 12 students take four different tests if three students are to take each test?

A. 369 6
B. 369 60
C. 369 000
D. 369 600

Q7. The number of ways in which a team of eleven players can be selected from 22 players including 2 of them and excluding 4 of them is

A. ${}^{16}C_{11}$

B. ${}^{16}C_{5}$

C. ${}^{16}C_{9}$

D. ${}^{20}C_{9}$

Q8. A bag contains six white marbles and five red marbles. Find the number of ways four marbles can be drawn from the bag if two must be white and two red

A. 100

B. 150

C. 300

D. 400

Q9. A bag contains six white marbles and five red marbles.Find the number of ways four marbles can be drawn from the bag if they must all be of the same colour.

A. 200
B. 210
C. 120
D. 20

Q10. Out of 12 employees, a group of four trainees is to be sent for software testing and QA training of one month.

What if two employees refuse to go together for training?

A. $10_{C_4} + 10_{C_3} + 10_{C_2}$
B. $10_{C_5} + 10_{C_3} + 10_{C_3}$
C. $10_{C_4} + 2.10_{C_3} + 10_{C_3}$
D. $10_{C_4} + 10_{C_3} + 10_{C_3}$

Q11. Out of 12 employees,a group of four trainees is to be sent for' software testing and QA ' trainingfor one month. Two employees want to go together that is either they both go or both do not go for training.

A. $10_{C_4} + 10_{C_1}$
B. $10_{C_4} + 10_2$
C. $10_{C_4} + 4$
D. $10_{C_4} + 20_{C_3}$

Q12. Out of 12 employees, a group of four trainees is to be sent for software testing and QA training of one month.Two employees want to go together and two employees refuse to go together.

A. $10_{C_4} + 10_{C_1}$
B. $10_{C_4} + 10_2$
C. $8_{C_4} + 8_{C_1}$
D. None of these

Q13. The minimum number of students needed to guarantee that five of them belong to the same class (freshman, sophomore, junior, senior)

A. Here the n=3 classes are the pigeonholes and k+1=7 so k=8. Thus, among any kn+1=14 student (pigeons), five of them belong to the same class.

B. Here the n=4 classes are the pigeonholes and k+1=5 so k=4. Thus, among any kn+1=17 student (pigeons), five of them belong to the same class.

C. Here the n=6 classes are the pigeonholes and k+1=50 so k=41. Thus, among any kn+1=107 student (pigeons), five of them belong to the same class.

D. Here the n=24 classes are the pigeonholes and k+1=15 so k=4. Thus, among any kn+1=17 student (pigeons), five of them belong to the same class.

Q14. Let L be a list (not necessarily in alphabetical order) of the 26 letters in the English alphabet (which consists of 5 vowels,A,E,I,O,U and 21 consonants) then L has a sublist consisting ……

A. of four or more consecutive consonants.

B. of five or more consecutive consonants

C. of six or more consecutive consonants

D. of eight or more consecutive consonants

Q15.Let L be a list (not necessarily in alphabetical order) of the 26 letters in the English alphabet (which consists of 5 vowels,A,E,I,O,U and 21 consonants). Assuming L begins with a vowel, say A, then L has a sublist consisting …….

A. of four or more consecutive consonants.

B. of five or more consecutive consonants

C. of six or more consecutive consonants

D. of eight or more consecutive consonants

Q16. The minimum number n of integers to be selected from S = {1, 2, ..., 9}so that the sum of two of the n integers is even is…….

A. 3

B. 4

C. 5

D. 6

Q17.The minimum number n of integers to be selected from S = {1, 2, ..., 9}so that the difference of two of the n integers is 5 is

A. 5

B. 6

C. 7

D. 8

Answer for Self Assessment

1.	A	2.	A	3.	A	4.	B	5.	C
6.	D	7.	C	8.	B	9.	D	10.	D
11.	B	12.	D	13.	B	14.	A	15.	B
16.	A	17.	B						

Review Questions

Q1. How many different permutations are there of the set{*a, b, c, d, e, f, g*}?

Q2. Let *S* = {1, 2, 3, 4, 5}.

a) List all the 3-permutations of *S*.

b) List all the 3-combinations of *S*.

Q3. Find the value of each of these quantities.

a) *C(*5, 1*)* **b)** *C(*5, 3*)*

b) *C(*8, 4*)* **d)** *C(*8, 8*)*C(8, 0) **f)** C(12, 6)

Q4. In how many different orders can five runners finish arace if no ties are allowed?

Q5. There are six different candidates for governor of a state.In how many different orders can the names of the candidatesbe printed on a ballot?

Q6. How many ways are there for 10 women and six mento stand in a line so that no two men stand next to eachother?

Q7. How many bit strings of length 12 contain**a)** exactly three 1s?**b)** at most three 1s?**c)** at least three 1s?**d)** an equal number of 0s and 1s?

Q8. In how many ways can a set of two positive integers lessthan 100 be chosen?

Q9. How many subsets with an odd number of elements does a set with 10 elements have?

Q10. How many bit strings of length 10 havea) exactly three 0s?b) more 0s than 1s?c) at least seven 1s?d) at least three 1s

Further Readings

1. Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
2. Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.
3. Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.
4. Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 07: Graphs Terminology

CONTENTS

Objectives
7.1 Graphs and Graph Models
7.2 Introduction , Data Structures:
7.3 Stacks, Queues, and Priority Queues
7.4 Multigraphs
7.5 Degree of a Vertex
Summary
Self Assessment
Answer for Self Assessment
Review Questions
Further Readings:

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- understand different types of graphs
- understand the degree of a graph
- understand the degree of a vertex in a graph.
- understand what is Handshaking theorem
- understand some special simple graphs.
- Understand what is a complete graph and a complete bipartite graph.
- Understand what is a union and subgraphs.
- Understand how to represent a graph in a matrix form.
- Understand what is an adjacency matrix
- Understand what is graph isomorphism

7.1 Graphs and Graph Models

We begin with the definition of a graph.

DEFINITION 1 A graph G = (V,E) consists of V, a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.

Remark: The set of vertices V of a graph G may be infinite. A graph with an infinite vertex set or an infinite number of edges is called an infinite graph, and in comparison, a graph with a finite vertex set and a finite edge set is called a finite graph. In this book, we will usually consider only finite graphs.

Now suppose that a network is made up of data centres and communication links between computers. We can represent the location of each data centre by apoint and eachcommunications link by a line segment, as shown in the Figure below. This computer network can be modelled using a graph in which the vertices of the graph represent the data centres and the edges represent communication links. In general, we visualize

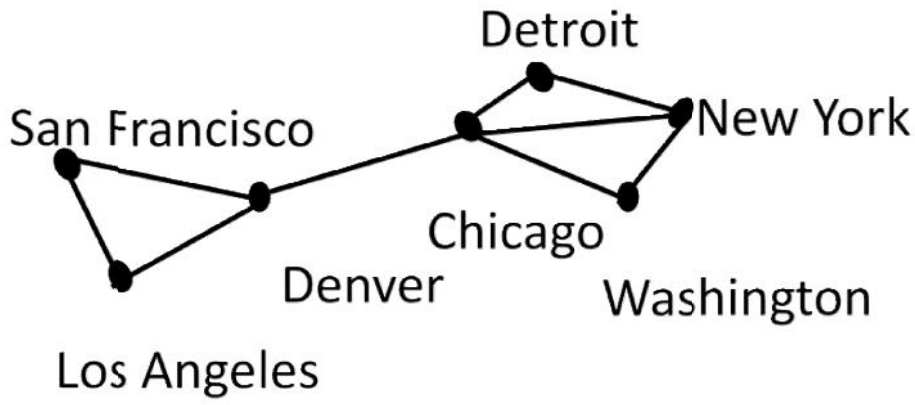
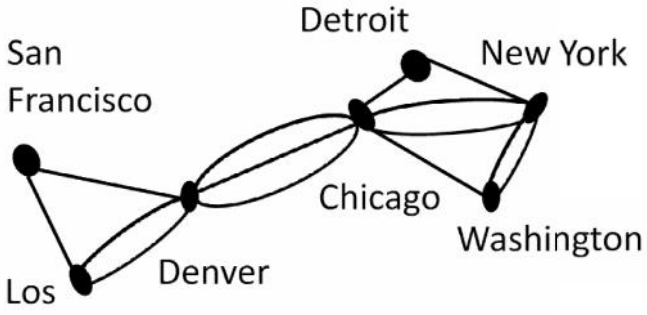


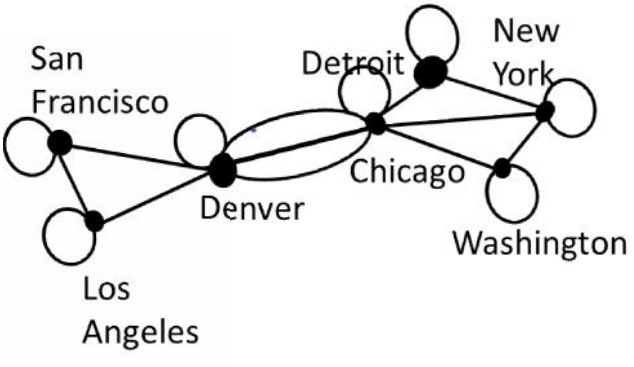
Figure A Computer Network

graphs by using points to represent vertices and line segments, possibly curved, to represent edges, where the endpoints of a line segment representing an edge are the points representing the endpoints of the edge.Whenwe draw a graph, we generally try to draw edges so that they do not cross. However, this is not necessary because any depiction using points to represent vertices and any form of connection between vertices can be used. Indeed, some graphs cannot be drawn in the plane without edges crossing. The key point is that the way we draw a graph is arbitrary, as long as the correct connections between vertices are depicted. Note that each edge of the graph representing this computer network connects two different vertices.That is, no edge connects a vertex to itself. Furthermore, no two different edges connect the same pair of vertices.A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a simple graph. Note that in a simple graph, each edge is associated to an unordered pair of vertices, and no other edge is associated to this same edge. Consequently, when there is an edge of a simple graph associated to {u,v}, we can also say, without possible confusion, that{u,v}is an edge of the graph. A computer network may contain multiple links between data centers, as shown in Figure below. To model such networks we need graphs that have more than one edge connecting the same pair of vertices. Graphs that may have multiple edges connecting the same vertices are called multigraphs.Whentherearemdifferentedgesassociatedtothesameunorderedpairofvertices {u,v}, we also say that {u,v} is an edge of multiplicity m. That is, we can think of this set of edges as m different copies of an edge{u,v}.



A Computer network with multiple links between data centers.

Sometimes a communications link connects a data center with itself, perhaps a feedback loop for diagnostic purposes. Such a network is illustrated in the Figure below.To model this network we



A Computer Network with Diagnostic Links

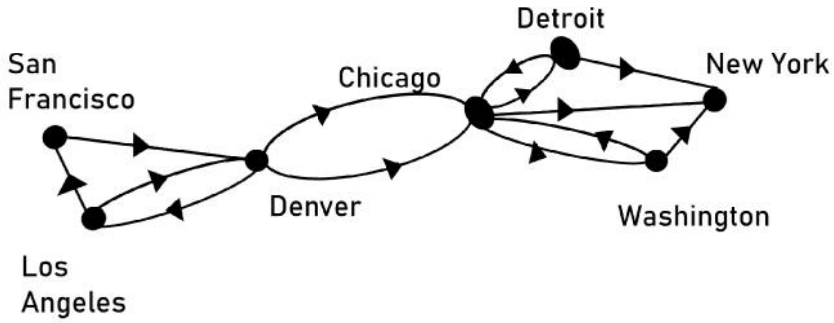


FIGURE: A Communications Network with One-Way Communications Links.

need to include edges that connect a vertex to itself. Such edges are called loops, and sometimes we may even have more than one loop at a vertex. Graphs that may include loops, and possibly multiple edges connecting the same pair of vertices or a vertex to itself, are sometimes called pseudographs. So far the graphs we have introduced are undirected graphs. Their edges are also said to be undirected. However, to construct a graph model, we may find it necessary to assign directions to the edges of a graph. For example, in a computer network, some links may operate in only one direction (such links are called single duplex lines). This may be the case if there is a large amount of traffic sent to some data centers, with little or no traffic going in the opposite direction. Such a network is shown in Figure above. To model such a computer network we use a directed graph. Each edge of a directed graph is associated with an ordered pair. The definition of the directed graph we give here is more general than the one we used earlier, where we used directed graphs to represent relations.

7.2 Introduction , Data Structures:

Graphs, directed graphs, trees and binary trees appear in many areas of mathematics and computer science.

However, in order to understand how these objects may be stored in memory and to understand algorithms on them, we need to know a little about certain data structures.

We assume the reader does understand linear and two-dimensional arrays; hence we will only discuss linked lists and pointers, and stacks and queues below.

Linked Lists and Pointers:

Linked lists and pointers will be introduced using an example. Suppose a brokerage firm maintains afile in which each record contains a customer's name and salesman; say the file contains the following data:

Customer	Adams	Brown	Clark	Drew	Evans	Farmer	Geller	Hiller	Infeld
Salesman	smith	Ray	Ray	Jones	Smith	Jones	Ray	Smith	ray

There are two basic operations that one would want to perform on the data:

Operation A: Given the name of a customer, find his salesman.

Operation B: Given the name of a salesman, find the list of his customers.

We discuss several ways the data may be stored in the computer and the ease with which one can performoperations A and B on the data.

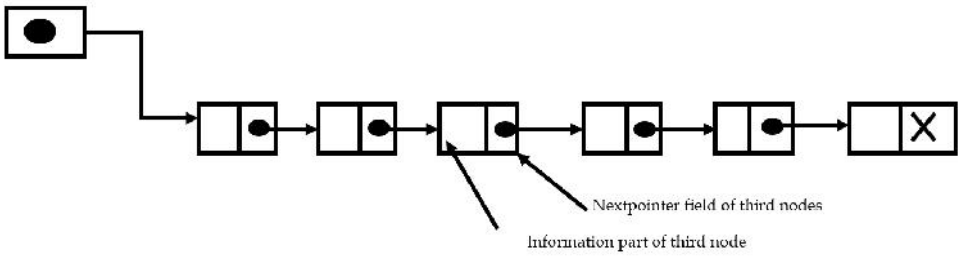
Clearly, the file could be stored in the computer by an array with two rows (or columns) of nine names.

Since the customers are listed alphabetically, one could easily perform operation A. However, in order to performoperation B one must search through the entire array.

One can easily store the data in memory using a two-dimensional array where, say, the rows correspond toan alphabetical listing of the customers and the columns correspond to an alphabetical listing of the salesmen,and where there is a 1 in the matrix indicating the salesman of a customer and there are 0's elsewhere. The maindrawback of such a representation is that there may be a waste of a lot of memory because many 0's maybe in thematrix. For example, if a firm has 1000 customers and 20 salesmen, one would need 20 000 memory locationsfor the data, but only 1000 of them would be useful.

We discuss below a way of storing the data in memory that uses linked lists and pointers. By a linked list,we mean a linear collection of data elements, called nodes, where the linear order is given by means of a fieldof pointers. Figure below is a schematic diagram of a linked list with six nodes. That is, each node is divided intotwo parts: the first part contains the information of the element (e.g., NAME, ADDRESS, ...), and the secondpart, called the link field or nextpointer field, contains the address of the next node in the list. This pointer fieldis indicated by an arrow drawn from one node to the next node in the list. There is also a variable pointer, calledSTART in Figure below, which gives the address of the first node in the list. Furthermore, the pointer field of the last

the node contains an invalid address, called a null pointer, which indicates the end of the list.



One main way of storing the original data pictured in Figure below, uses linked lists. Observe that there are separate(sorted alphabetically) arrays for the customers and the salesmen. Also, there is a pointer array SLSM parallelto CUSTOMER which gives the location of the salesman of a customer, hence operation A can be performedvery easily and quickly. Furthermore, the list of customers of each salesman is a linked list as discussed above.

Specifically, there is a pointer array START parallel to SALESMAN which points to the first customer of asalesman, and there is an array NEXT which points to the location of the next

customer in the salesman's list(or contains a 0 to indicate the end of the list). This process is indicated by the arrows in Figure below for the

salesman Ray.

Customer		SLSM	Next
1	Adams	3	5
2	Brown	2	3
3	Clark	2	7
4	Drew	1	6
5	Evans	3	8
6	Farmer	1	0
7	Geller	2	9
8	Hill	3	0
9	Infeld	2	0

Salesman	Start	
Jones	4	1
Ray	2	2
Smith	1	3

Operation B can now be performed easily and quickly; that is, one does not need to search through the list of all customers in order to obtain the list of customers of a given salesman. Figure above gives such an algorithm (which is written in pseudo-code).

7.3 Stacks, Queues, and Priority Queues

There are data structures other than arrays and linked lists that will occur in our graph algorithms. These structures, stacks, queues, and priority queues, are briefly described below.

Stack: A stack, also called a last-in-first-out (LIFO) system, is a linear list in which insertions and deletionscan take place only at one end, called the "top" of the list. This structure is similar in its operation to a stackof dishes on a spring system, as pictured in Figure (a) below. Note that new dishes are inserted only at the top ofthe stack and dishes can be deleted only from the top of the stack.

Queue: A queue, also called a first-in-first-out (FIFO) system, is a linear list in which deletions can only takeplace at one end of the list, the "front" of the list, and insertions can only take place at the other end of thelist, the "rear" of the list. The structure operates in much the same way as a line of people waiting at a busstop, as pictured in Figure (b) below. That is, the first person in line is the first person to board the bus, and a newperson goes to the end of the line.

Priority queue: Let S be a set of elements where new elements may be periodically inserted, but where the current largest element (element with the "highest priority") is always deleted. Then S is called a priority queue. The rules "women and children first" and "age before beauty" are examples of priority queues. Stacks

and ordinary queues are special kinds of priority queues. Specifically, the element with the highest priority in a stack is the last element inserted, but the element with the highest priority in a queue is the first element inserted.

(a) Stack of dishes. (b) Queue waiting for a bus.

DEFINITION 2 A directed graph (or digraph) (V,E) consists of a nonempty set of vertices V and a set of directed edges (or arcs) E. Each directed edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (u, v) is said to start at u and end at v.

When we depict a directed graph with a line drawing, we use an arrow pointing from u to v to indicate the direction of an edge that starts at u and ends at v.A directed graph may contain loops anditmaycontainmultipledirectededgesthatstartandendatthesamevertices.Adirectedgraph may also contain directed edges that connect vertices u and v in both directions; that is,when a digraphcontainsanedgefromutov,itmayalsocontainoneormoreedgesfromv tou.Notethat weobtainadirectedgraphwhenweassignadirectiontoeachedgeinanundirectedgraph.When a directed graph has no loops and has no multiple directed edges, it is called a simple directed graph. Because a simple directed graph has at most one edge associated to each ordered pair of vertices (u, v), we call (u, v) an edge if there is an edge associated to it in the graph. In some computer networks, multiple communication links between two data centers may be present. Directed graphs that may have multiple directed edges from a vertex to a second (possibly the same) vertex are used to model such networks.We called such graphs directed multigraphs. When there are m directed edges, each associated to an ordered pair of vertices (u, v), we say that (u, v) is an edge of multiplicity m.

TABLE Graph Terminology.

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simplegraph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudo graph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

For some models we may need a graph where some edges are undirected, while others are directed. A graph with both directed and undirectededgesis called a mixedgraph. For example, a mixed graph might be used to model a computer network containing links that operate in both directions and other links that operate only in one direction. This terminology for the various types of graphs is summarized in Table above. We will sometimes use the term graph as a general term to describe graphs with directed or undirected edges (or both), with or without loops, and with or without multiple edges. At other times, when the context is clear, we will use the term graph to refer only to undirected graphs. Becauseoftherelativelymoderninteresting graphtheory,andbecause it hasapplicationstoa wide variety of disciplines, many different terminologies of graph theory have been introduced. The reader should determine how such terms are being used whenever they are encountered.

Theterminologyusedbymathematicianstodescribegraphshasbeenincreasinglystandardized, but the terminology used to discuss graphs when they are used in other disciplines is still quite varied.

Although the terminology used to describe graphs may vary, three key questions can help us understand the structure of a graph:

Are the edges of the graph undirected or directed (or both)? Ifthegraphisundirected,aremultipleedgespresentthatconnectthesamepairofvertices? If the graph is directed, are multiple directed edges present? Are loops present?

Answering such questions helps us understand graphs. It is less important to remember the particular terminology used.

A graph G consists of two things:

(i) A set V = V (G) whose elements are called vertices, points, or nodes of G.

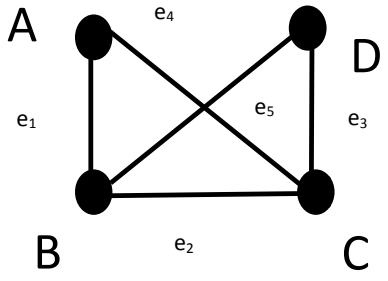
(ii) A set E = E(G) of unordered pairs of distinct vertices called edges of G.

We denote such a graph by G(V, E) when we want to emphasize the two parts of G Vertices u and v are said to be adjacent or neighbours if there is an edge e = {u, v}. In such a case, u and v are called the endpoints of e, and e is said to connect u and v. Also, the edge e is said to be incident on each of its

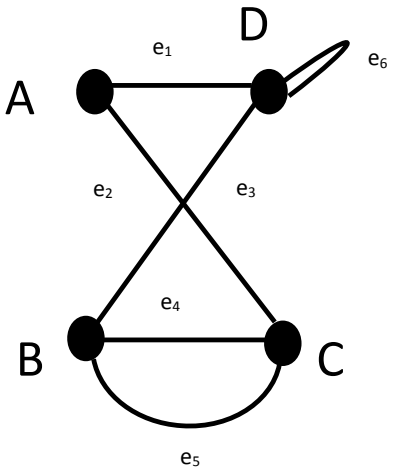
endpoints u and v. Graphs are pictured by diagrams in the plane in a natural way. Specifically, each vertex v in V is represented by a dot (or small circle), and each edge e = {v1, v2} is represented by a curve that connects its endpoints v1 and v2 For example, the figure represents the graph G(V, E) where:

(i) V consists of vertices A, B, C, D.

(ii) E consists of edges e1 = {A, B}, e2= {B, C}, e3 = {C, D}, e4 = {A, C}, e5 = {B, D}

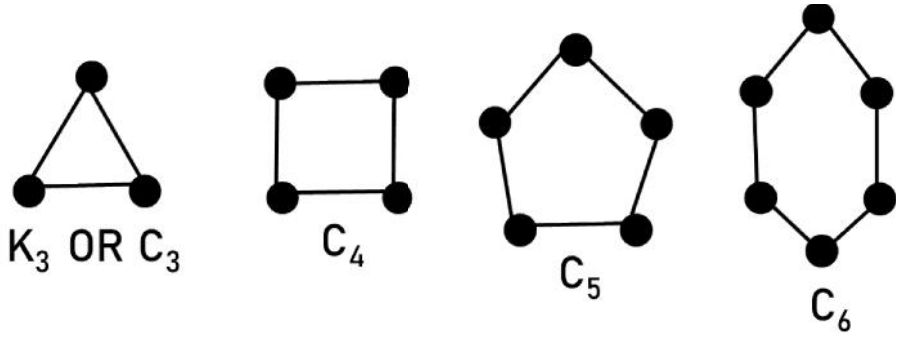


Graph

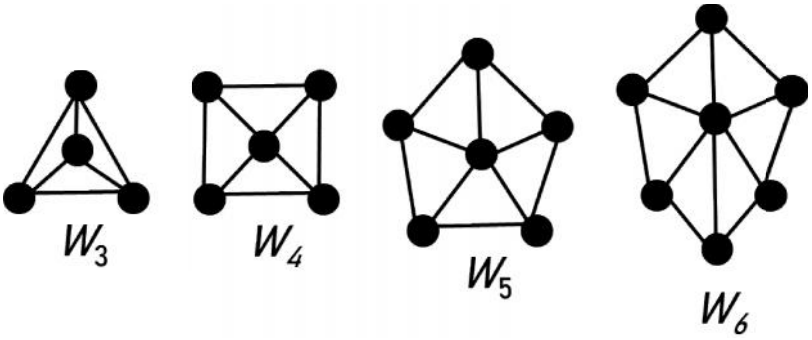


Multigraph

A cycle C_n, $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, . . . , $\{v_{n-1}, v_n\}$, and $\{v_n, v_1\}$. The cycles C_3, C_4, C_5, and C_6 are displayed in Figure below.

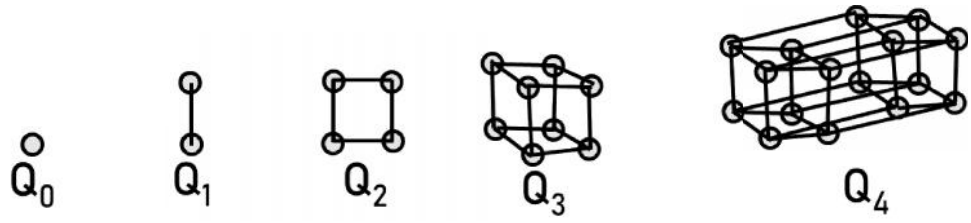


We obtain a wheel W_n when we add an additional vertex to a cycle C_n, for $n \geq 3$, and connect this new vertex to each of the n vertices in C_n, by new edges. The wheels W_3, W_4, W_5, and W_6 are displayed in the Figure below.

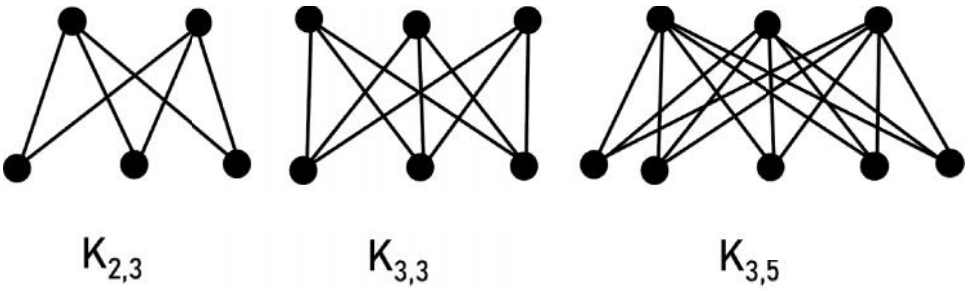


For any $n \in N$, the hypercube Q_n is a simple graph consisting of two copies of Q_{n-1}connected together at corresponding nodes.

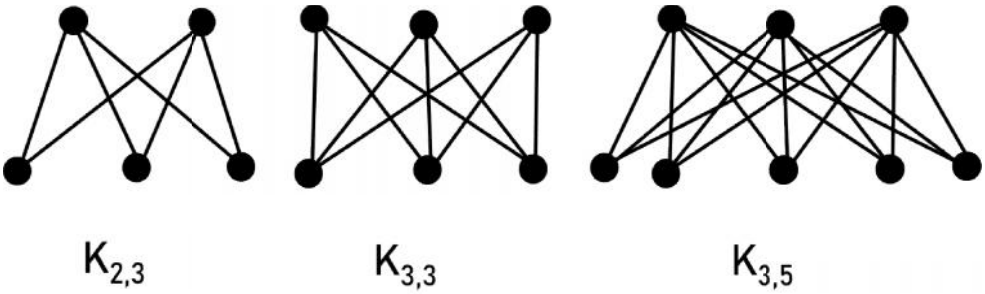
Q_0 has 1 node.



The complete bipartite graphs $K_{2,3}$, $K_{3,3}$, and $K_{3,5}$are displayed in the Figure below.

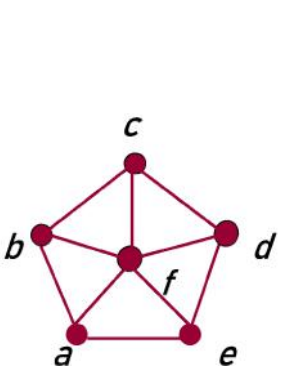


A complete bipartite graph $K_{m,n}$ is a graph that has its vertex set partitioned into two subsets of m and n vertices, respectively with an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

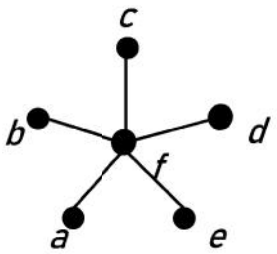


The union of 2 simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V = V_1 \cup V_2$ and edge set $E = E_1 \cup E_2$. The union is denoted by $G_1 \cup G_2$.

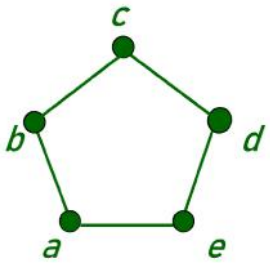
W_5 is the union of S_5 and C_5



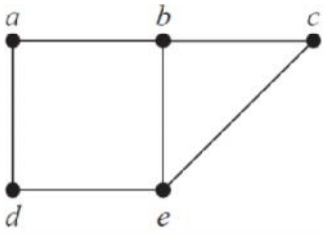
W_5



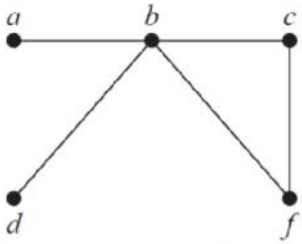
S_5



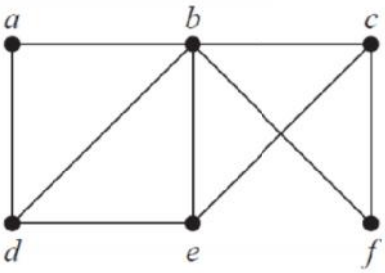
C_5



G_1



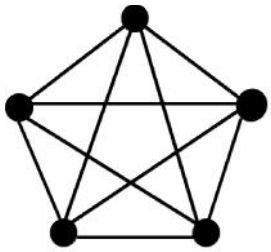
G_2



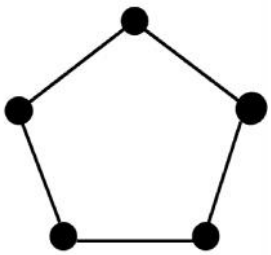
$G_2 \cup G_2$

A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$ and $F \subseteq E$.

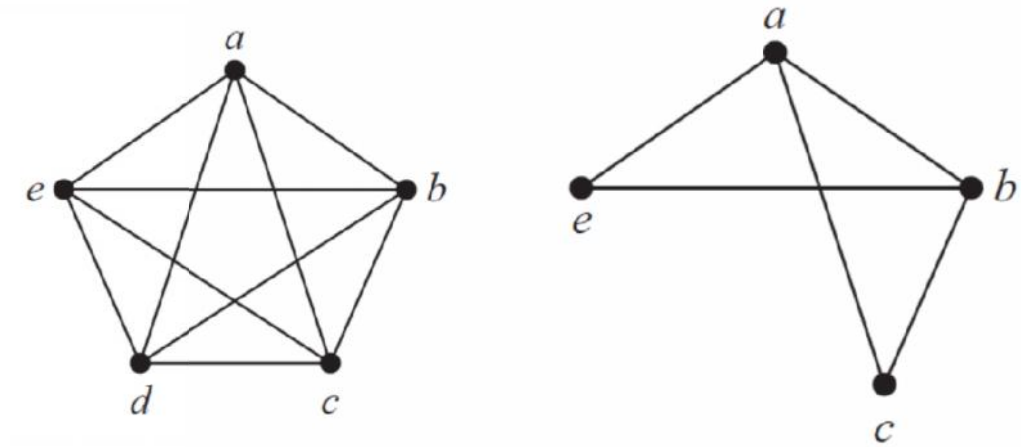
C_5 is a subgraph of K_5



K_5



C_5



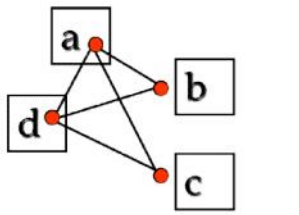
A Subgraph of K_5

Let G = (V, E) be a simple graph with |V| = n. Suppose that the vertices of G are listed in arbitrary order as $v_1, v_2, \ldots, v_n$.

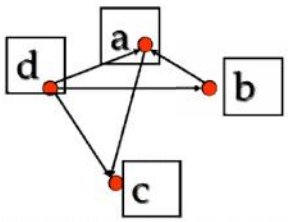
The adjacency matrix A (or A_G) of G, with respect to this listing of the vertices, is the n×n zero-one matrix with 1 as its (i, j) entry when v_i and v_j are adjacent, and 0 otherwise.

In other words, for an adjacency matrix A = [a_{ij}],

$a_{ij} = 1$ if $\{v_i, v_j\}$ is an edge of G,
$a_{ij} = 0$ otherwise.



Vertex	Adjacent Vertices
a	b, c, d
b	a, d
c	a, d
d	a, b, c



Initial Vertex	Terminal Vertices
a	c
b	a
c	
d	a, b, c

7.4 Multigraphs

Consider the diagram in figure below. The edges e4 and e5 are called multiple edges since they connect thesame endpoints, and the edge e6 is called a loop since its endpoints are the same vertex. Such a diagram is calleda multigraph; the formal definition of a graph permits neither multiple edges nor loops. Thus a graph may bedefined to be a multigraph without multiple edges or loops.

Remark: Some texts use the term graph to include multigraphs and use the term simple graph to mean a graphwithout multiple edges and loops.

7.5 Degree of a Vertex

The degree of a vertex v in a graph G, written deg (v), is equal to the number of edges in G which containv, that is, which are incident on v. Since each edge is counted twice in counting the degrees of the vertices of G,we have the following simple but important result.

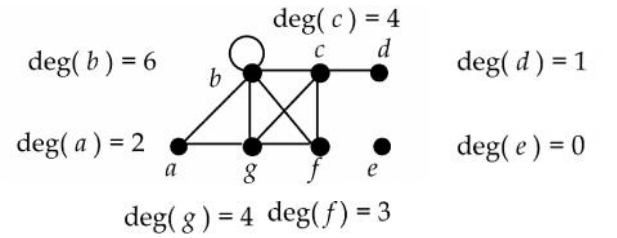
Handshaking Theorem

Theorem :The sum of the degrees of the vertices of a graph G is equal to twice the number of edges in G.

Q. Find the degree of all the other vertices.

deg(a) = 2, deg(c) = 4, deg(f) = 3, deg(g) = 4

TOTAL of degrees = 2 + 4 + 3 + 4 + 6 + 1 + 0 = 20



A vertex is said to beeven or odd according as its degree is an even or an odd number. Thus A and D are even vertices whereas B andC are odd vertices.

The theorem also holds for multigraphs where a loop is counted twice toward the degree of its endpoint. Forexample, we have deg(D) = 4 since the edge e6 is counted twice; hence D is an even vertex.

A vertex of degree zero is called an isolated vertex.

Finite Graphs, Trivial Graph

A multigraph is said to be finite if it has a finite number of vertices and a finite number of edges. Observe that

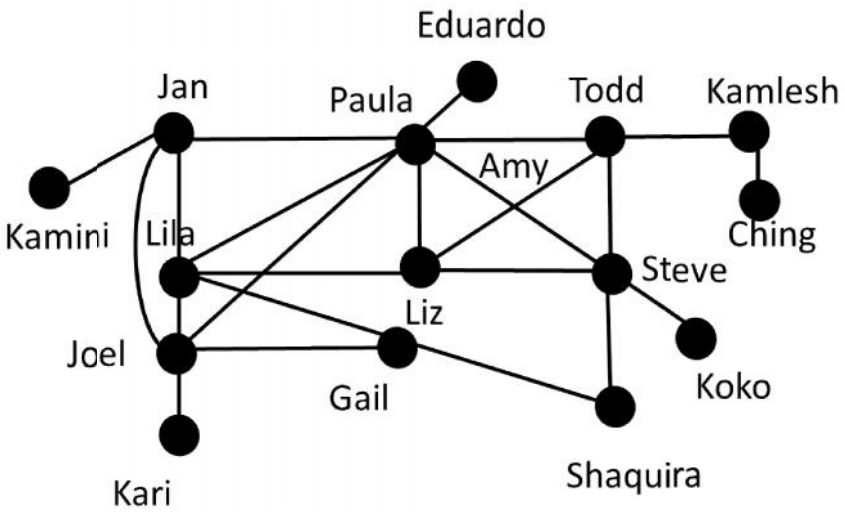
a graph with a finite number of vertices must automatically have a finite number of edges and so must be finite.

Graph Models

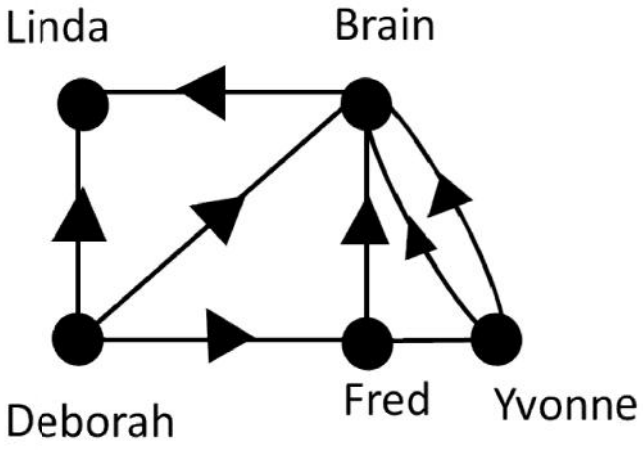
Graphsareusedinawidevarietyofmodels.Webeganthissectionbydescribinghowtoconstruct graph models of communications networks linking data centers. We will complete this section by describing some diverse graph models for some interesting applications. We will return to many of these applications later in this chapter. We will introduce additional graph models in subsequent sections of this and later chapters. Also, recall that directed graph models for some applications were introduced earlier. When we build a graph model, we need to make sure that we have correctly answered the three key questions we posed about the structure of a graph.

Social Networks

Graphs are extensively used to model social structures based on different kinds of relationships between people or groups of people. The socialstructures, andthe graphs that represent them, are known as social networks. In these graph models, individuals or organizations are represented by vertices; relationships between individuals or organizations are represented by edges. The study of social networks is an extremely active multidisciplinary area, and many different types of relationships between people have been studied using them



An Acquaintanceship graph



An influence graph.

We will introduce some of the most commonly studied social networks here. More information about social networks can be found in [Ne10] and [EaKl10].

EXAMPLE 1 Acquaintanceship and Friendship Graphs We can use a simple graph to represent whether two people know each other, that is, whether they are acquainted, or whether they are friends (either in the real world in the virtual world via a social networking site such as Facebook). Each person in a particular group of people is represented by a vertex. An undirected edge is used to connect two people when these people know each other when we are concerned only with acquaintanceship, or whether they are friends. No multiple edges and usually no loops are used. (If we want to include the notion of self-knowledge, we would include loops.) A small acquaintanceship graph is shown in the Figure above. The acquaintanceship graph of all people in the world has more than six billion vertices and probably more than one trillion edges! We will discuss this graph further in the coming section ▲

EXAMPLE 2 Influence Graphs In studies of group behaviour it is observed that certain people can influence the thinking of others. A directed graph called an influence graph can be used to model this behaviour. Each person of the group is represented by a vertex. There is a directed edge from vertexa tovertexb whenthepersonrepresentedbyvertexa caninfluencethepersonrepresented by vertex b. This graph does not contain loops and it does not contain multiple directed edges. An example of an influence graph for members of a group is shown in Figure above. In the group modelled by this influence graph, Deborah cannot be influenced, but she can influence Brian, Fred, and Linda. Also,Yvonne and Brian can influence each other. ▲

EXAMPLE 3 Collaboration Graphs A collaboration graph is used to model social networks where two people are related by working together in a particular way. Collaboration graphs are simple graphs, as edges in these graphs are undirected and there are no multiple edges or loops.Vertices in these graphs represent people; two people are connected by an undirected edge when the peoplehavecollaborated.Therearenoloopsnormultipleedgesinthesegraphs.TheHollywood graph is a collaborator graph that represents actors by vertices and connects two actors with an edge if they have worked together on a movie or television show. The Hollywood graph is a huge graph with more than 1.5 million vertices (as of early 2011).We will discuss some aspects of the Hollywood graph later. In an academic collaboration graph, vertices represent people (perhaps restricted to members of a certain academic community) and edges link two people if they have jointly published a paper. The collaboration graph for people who have published research papers in mathematics was found in 2004 to have more than 400,000 vertices and 675,000 edges, and these numbers have grown considerably since then. We will have more to say about this graph. Collaboration graphs have also been used in sports, where two professional athletes are considered to have collaborated if they have ever played on the same team during a regular season of their sport

.

COMMUNICATIONNETWORKS Wecanmodeldifferentcommunicationsnetworksusing vertices to represent devices and edges to represent the particular type of communications links of interest. We have already modelled a data network in the first part of this section.

EXAMPLE 4 Call Graphs Graphs can be used to model telephone calls made in a network, such as a long-distancetelephonenetwork.Inparticular,adirectedmultigraphcanbeusedtomodelcallshere each telephone number is represented by a vertex and each telephone call is represented by a directed edge. The edge representing a call starts at the telephone number from which the call was made and ends at the telephone number to which the call was made.We need directed edges becausethedirectioninwhichthecallismadematters.Weneedmultipledirectededgesbecause we want to represent each call made from a particular telephone number to a second number. A small telephone call graph is displayed in Figure

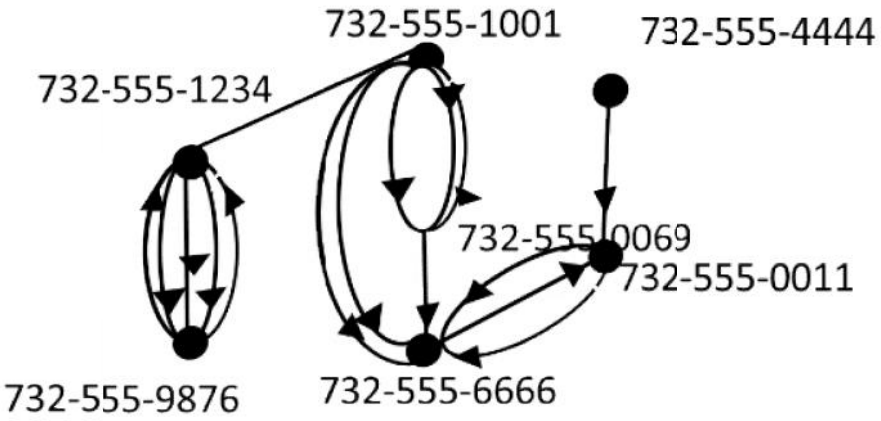
(a), representing seven telephone numbers. This graph shows, for instance, that three calls have been made from 732-555-1234 to 732-555-9876 and two in the other direction, but no calls have been made from 732-555-4444 toanyoftheothersixnumbersexcept732-555-0011.Whenwecareonlywhethertherehasbeen a call connecting two telephone numbers, we use an undirected graph with an edge connecting telephone numbers when there has been a call between these numbers. This version of the call graph is displayed in Figure

(b). Call graphs that model actual calling activities can be huge. For example, one call graph studiedatAT&T,whichmodelscallsduring20days,hasabout290millionverticesand4billion edges. We will discuss call graphs further. ▲

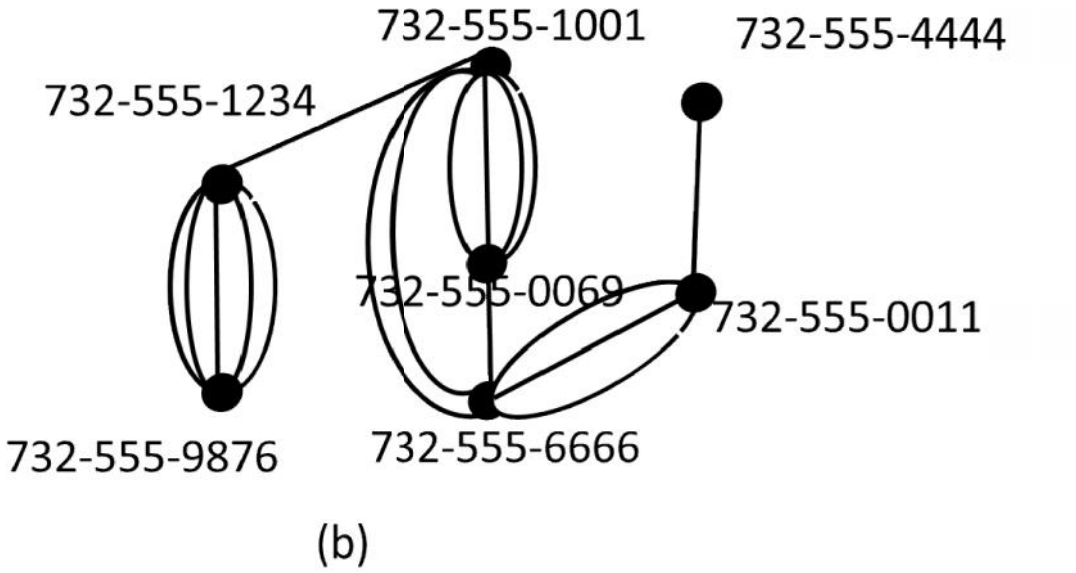
INFORMATION NETWORKS Graphs can be used to model various networks that link particular types of information. Here, we will describe how to model the Worldwide Web using a graph. We will also describe how to use a graph to model the citations in different types of documents.

EXAMPLE 5 The Web Graph The World Wide Web can be modelled as a directed graph where each Web page is represented by a vertex and where an edge starts at the Web page a and ends at the Web page b if there is a link on pointing to b. Because new Web pages are created and others removed somewhere on the Web almost every second, the Web graph changes on an almost continual basis. Many people are studying the properties of the Web graph to better understand the nature of the Web. We will return to Web graphs in later Chapters we will explain how theWeb graph is used by theWeb crawlers that search engines use to create indexes of Web pages. ▲

EXAMPLE 6 Citation Graphs Graphs can be used to represent citations in different types of documents, including academic papers, patents, and legal opinions. In such graphs, each document is represented by a vertex, and there is an edge from one document to a second document if the



(a)



first document cites the second in its citation list. (In an academic paper, the citation list is the bibliography or list of references; in a patent, it is the list of previous patents that are cited; and in a legal opinion it is the list of previous opinions cited.) A citation graph is a directed graph without loops or multiple edges. ▲

SOFTWARE DESIGN APPLICATIONS Graph models are useful tools in the design of the software. We will briefly describe two of these models here. EXAMPLE 7 ModuleDependencyGraphs Oneofthemostimportanttasksindesigningsoftwareishowto structureaprogramintodifferentparts,ormodules.Understandinghowthedifferentmodulesof a program interact is essential not only for program design but also for testing and maintenance oftheir sultingsoftware.Amoduledependencygraphprovidesausefultoolforunderstanding how different modules of a program interact. In a program dependency graph, each module is represented by a vertex.There is a directed edge from a module to a second module if the second module depends on the first. An example of a program dependency graph for a web browser is shown in the Figure below. ▲

EXAMPLE 8 Precedence Graphs and Concurrent Processing Computer programs can be executed more rapidly by executing certain statements concurrently. It is important not to execute a statement that requires results of statements not yet executed. The dependence of statements on previous statements can be represented by a directed graph. Each statement is represented by a vertex, and there is an edge from one statement to a second statement if the second statement cannot be executed before the first statement. This resulting graph is called a precedence graph. A computer program and its graph are displayed in the Figure below. For instance, the graph shows that statement S5 cannot be executed before statements S1, S2, and S4 are executed. ▲

TRANSPORTATION NETWORKS We can use graphs to model many different types of transportation networks, including road, air, and rail networks, as well shipping networks.

EXAMPLE 9 Airline Routes We can model airline networks by representing each airport by a vertex. In particular, we can model all the flights by a particular airline each day using a directed edge to represent each flight, going from the vertex representing the departure airport to the vertex representing the destination airport.The resulting graph will generally be a directed multigraph, as there may be multiple flights from one airport to some other airport during the same day. ▲

EXAMPLE 10 Road Networks Graphs can be used to model road networks. In such models, vertices represent intersections and edges represent roads. When all roads are two-way and there is at most one road connecting two intersections, we can use a simple undirected graph to model the road network. However, we will often want to model road networks when some roads are one-way and when there may be more than one road between two intersections. To build such models, we use undirected edges to represent two-way roads and we use directed edges to represent

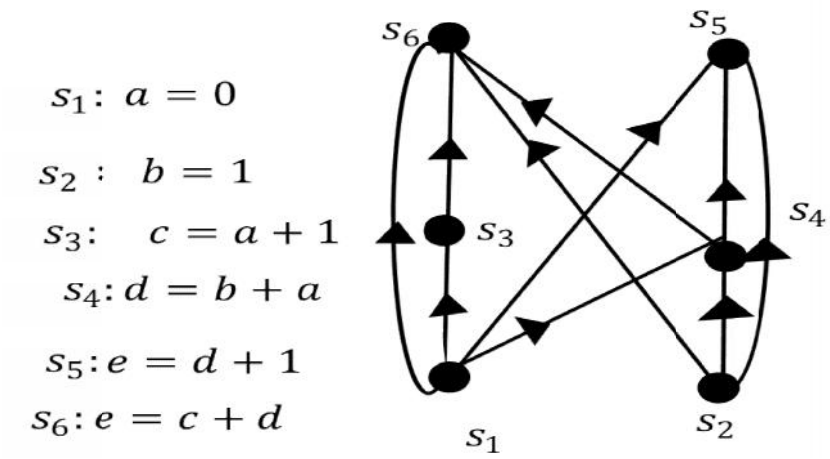


Figure A precedence graph

one-way roads. Multiple undirected edges represent multiple two-way roads connecting the same two intersections. Multiple directed edges represent multiple one-way roads that start at one intersection and end at a second intersection. Loops represent loop roads. Mixed graphs are needed to model road networks that include both one-way and two-way roads. ▲

BIOLOGICAL NETWORKS Many aspects of the biological sciences can be modelled using graphs.

EXAMPLE 11 Niche Overlap Graphs in Ecology Graphs are used in many models involving the interaction of different species of animals. For instance, the competition between species in an ecosystem can be modelled using a niche overlap graph. Each species is represented by a vertex. An undirected edge connects two vertices if the two species represented by these vertices compete (that is, some of the food resources they use are the same). A niche overlap graph is a simple graph because no loops or multiple edges are needed in this model. The graph in the Figure below models the ecosystem of a forest. We see from this graph that squirrels and raccoons compete but that crows and shrews do not. ▲

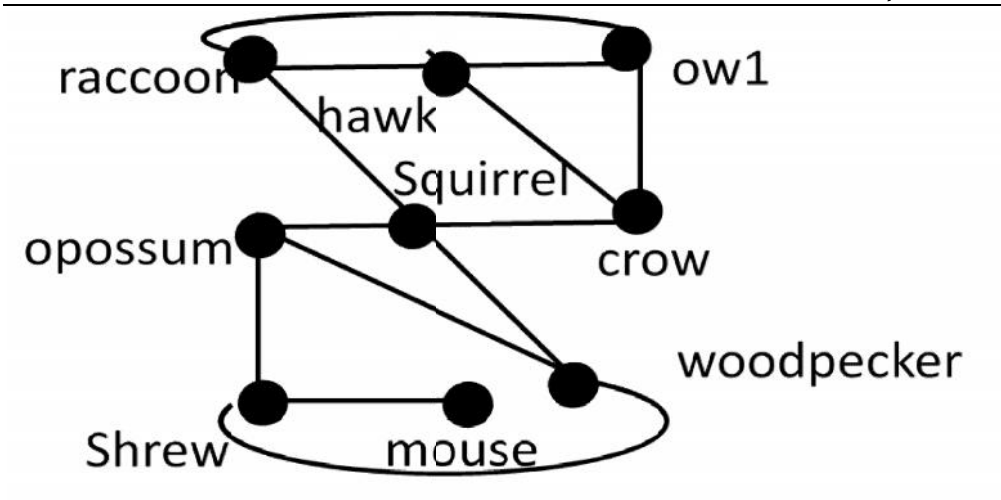
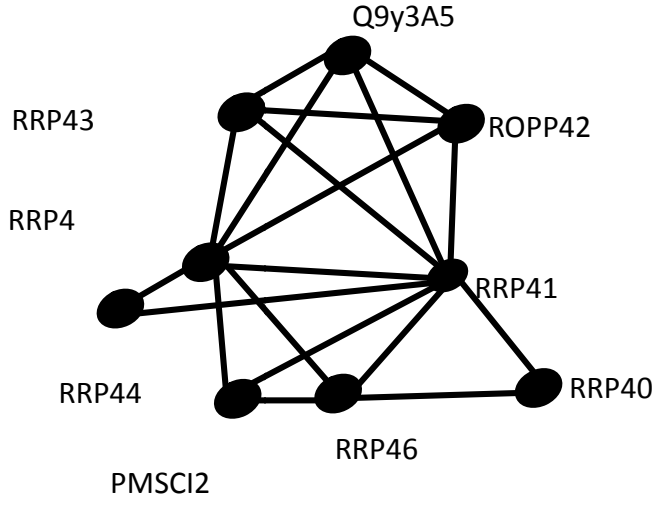
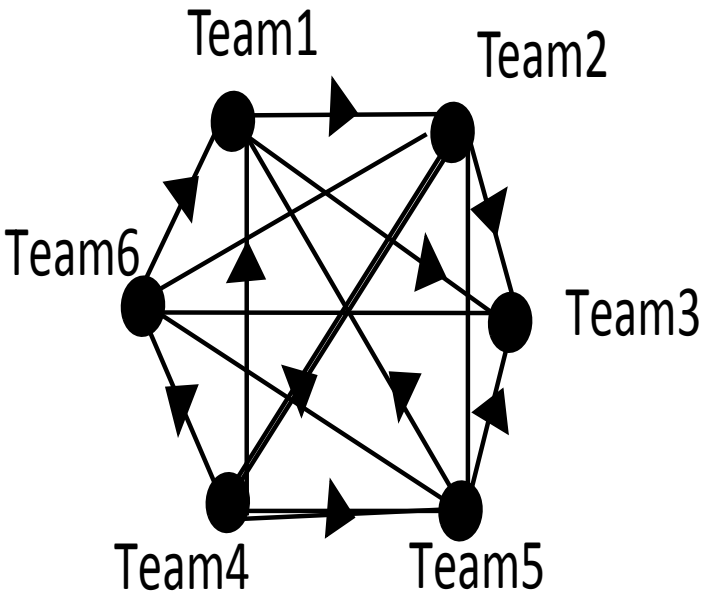


Figure A niche overlap graph

EXAMPLE 12 Protein Interaction Graphs A protein interaction in a living cell occurs when two or more proteins in that cell bind to perform a biological function. Because protein interactions are crucial for most biological functions, many scientists work on discovering new proteins and understanding interactions between proteins. Protein interactions within a cell can be modelled using a protein interaction graph (also called a protein-protein interaction network), an undirected graph in which each protein is represented by a vertex, with an edge connecting the vertices representing each pair of proteins that interact. It is a challenging problem to determine genuine protein interactions in a cell, as experiments often produce false positives, which conclude that two proteins interact when they really do not. Protein interaction graphs can be used to deduce important biological information, such as by identifying the most important proteins for various functions and the functionality of newly discovered proteins. Because there are thousands of different proteins in a typical cell, the protein interaction graph of a cell is extremely large and complex. For example, yeast cells have more than 6,000 proteins, and more than 80,000 interactions between them are known, and human cells have more than 100,000 proteins, with perhaps as many as 1,000,000 interactions between them. Additional vertices and edges are added to a protein interaction graph when new proteins and interactions between proteins are discovered. Because of the complexity of protein interaction graphs, they are often split into smaller graphs called modules that represent groups of proteins that are involved in a particular function of a cell. Figure 12 illustrates a module of the protein interaction graph described in [Bo04], comprising the complex of proteins that degrade RNA in human cells. To learn more about protein interaction graphs, see [Bo04], [Ne10], and [Hu07].





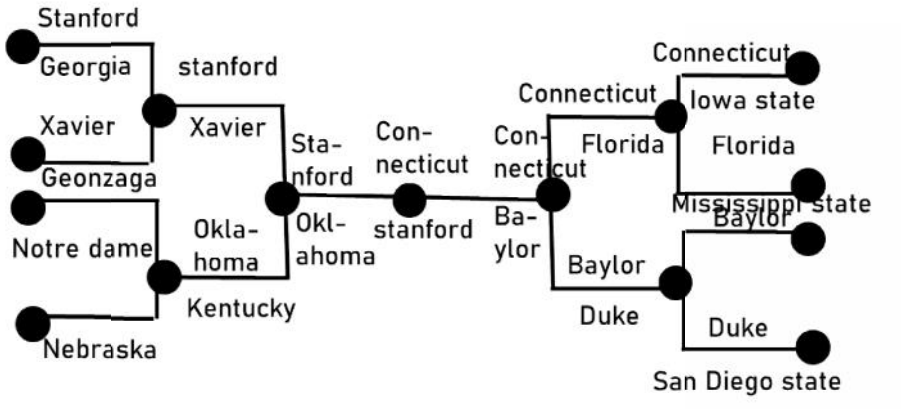


FIGURE A graph of a model of a round –robin tournament

TOURNAMENTS We now give some examples that show how graphs can also be used to model different kinds of tournaments.

EXAMPLE 13 Round-Robin Tournaments A tournament where each team plays every other team exactly once and no ties are allowed is called a round-robin tournament. Such tournaments can be modelled using directed graphs where each team is represented by a vertex. Note that (a,b) is an edge if team a beats team b. This graph is a simple directed graph, containing no loops or multiple directed edges (because no two teams play each other more than once). Such a directed graph model is presented in Figure 13. We see that Team 1 is undefeated in this tournament, and Team 3 is winless. ▲

EXAMPLE 14 Single-EliminationTournaments Atournamentwhereeachcontestantiseliminatedafterone loss is called a single-elimination tournament. Single-elimination tournaments are often used in sports, including tennis championships and the yearly NCAA basketball championship. We canmodelsuchatournamentusingavertextorepresenteachgameandadirectededgetoconnect a game to the next game the winner of this game played in. The graph in Figure below

represents the games played by the final 16 teams in the 2010 NCAA women's basketball tournament. ▲

SUBGRAPHS, ISOMORPHIC AND HOMEOMORPHIC GRAPHS

This section will discuss important relationships between graphs.

Subgraphs

Consider a graph G = G(V, E). A graph H = H (V, E) is called a subgraph of G if the vertices and edges of H are contained in the vertices and edges of G, that is if $V \subseteq V$ and $E \subseteq E$. In particular:

(i) A subgraph H (V, E) of G(V, E) is called the subgraph induced by its vertices V if its edge set E contains all edges in G whose endpoints belong to vertices in H

(ii) If v is a vertex in G, then G − v is the subgraph of G obtained by deleting v from G and deleting all edges in G which contain v.

(iii) If e is an edge in G, then G − e is the subgraph of G obtained by simply deleting the edge e from G.

Isomorphic Graphs

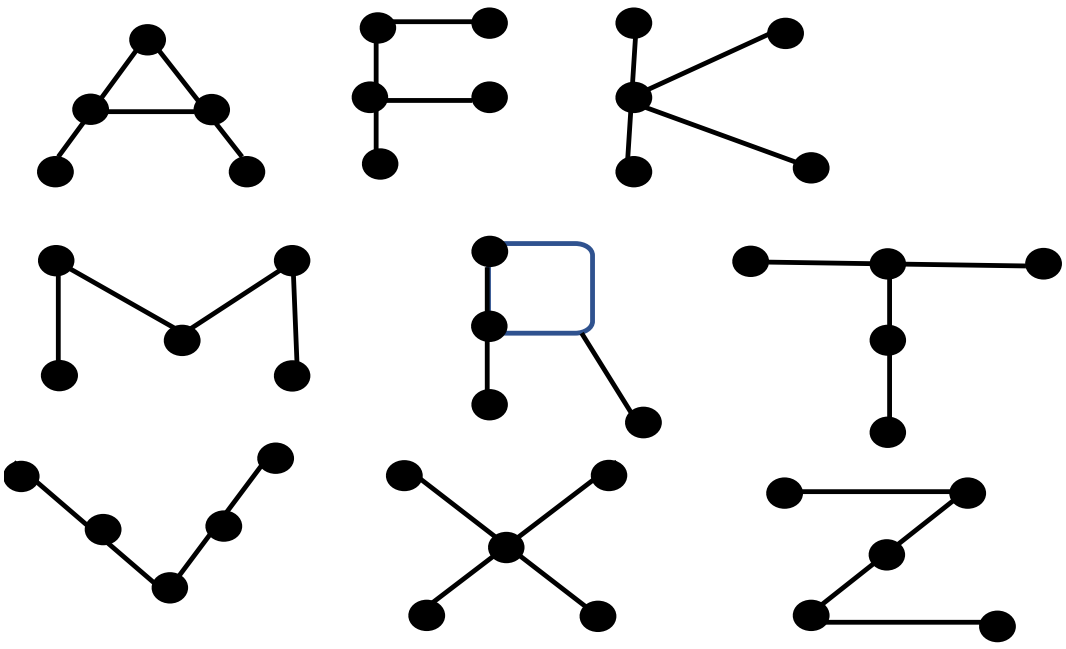
Graphs G(V, E) and G(V ∗, E∗) are said to be isomorphic if there exists a one-to-one correspondence

f: V → V ∗ such that {u, v} is an edge of G if and only if {f (u), f (v)} is an edge of G∗. Normally, we do not

distinguish between isomorphic graphs (even though their diagrams may "look different"). Figure 8-6 gives ten

graphs pictured as letters. We note that A and R are isomorphic graphs. Also, F and T are isomorphic graphs, K

and X are isomorphic graphs and M, S, V, and Z are isomorphic graphs.



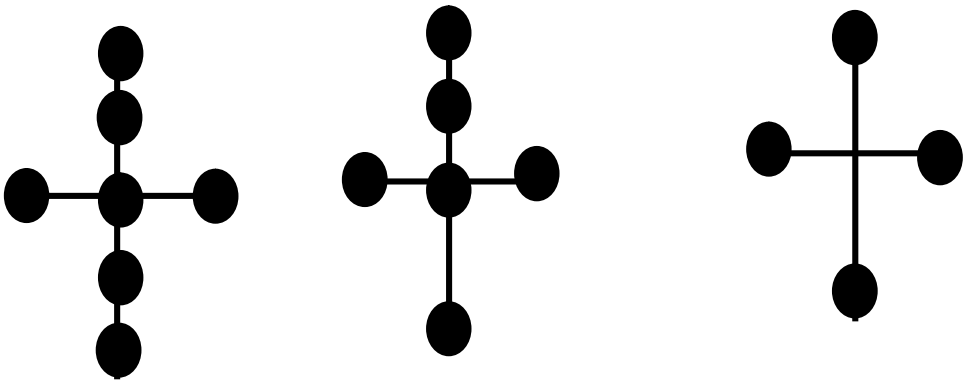
Homeomorphic Graphs

Given any graph G, we can obtain a new graph by dividing an edge of G with additional vertices. Two graphs

G and G∗ are said to homeomorphic if they can be obtained from the same graph or isomorphic graphs by this

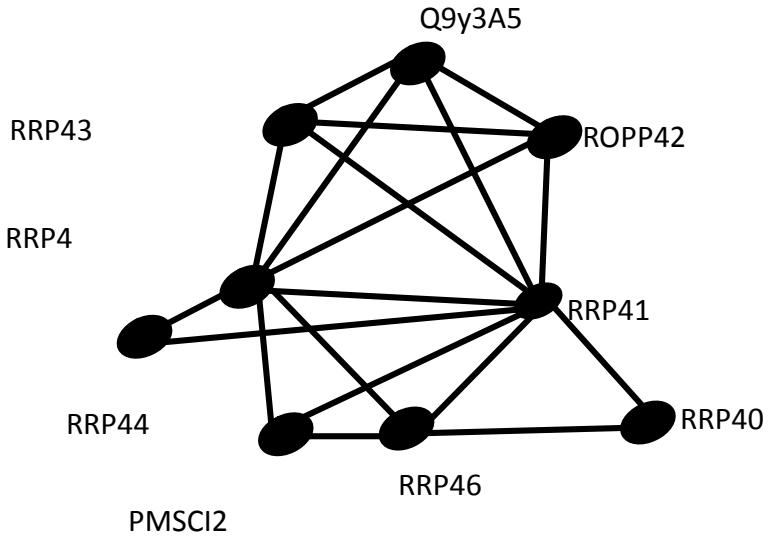
method. The graphs (a) and (b) in Figure below are not isomorphic, but they are homeomorphic since they can be

obtained from graph (c) by adding appropriate vertices.

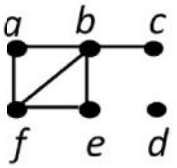


Summary

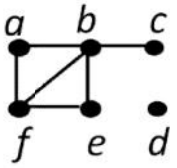
- We have learned different types of graphs
- We have learned the degree of a graph
- We have learned the degree of a vertex in a graph.
- We have learned what is Handshaking theorem
- We have learned some special simple graphs.
- We have learned what is a complete graph and a complete bipartite graph.
- We have learned what is a union and subgraphs.
- We have learned how to represent a graph in a matrix form.
- We have learned what is the adjacency matrix
- We have learned what is graph isomorphism



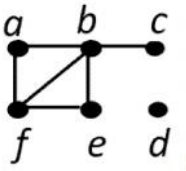
Self Assessment

Q1. There are how many vertices here 

A. 1
B. 3
C. 6
D. 9

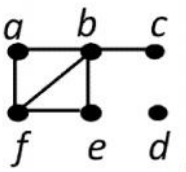
Q2. There are how manyedges here 

A. 1
B. 3
C. 6
D. 9

Q3. In the graph 

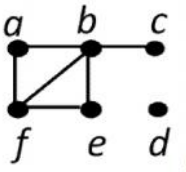
thedeg(a) =

A. 1
B. 2
C. 6
D. 9

Q4. In the graph 

thedeg(b) =

A. 1
B. 2
C. 4
D. 5

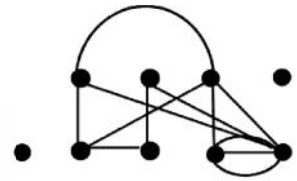
Q5. In the graph 

the sum of the degrees is

A. 4
B. 8

C. 10

D. 12



Q6. In the graph the sum of the degrees is

A. 4

B. 8

C. 20

D. 24

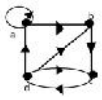
Q7. A simple graph exist with 15 vertices each of degree ……..

A. 5

B. 7

C. 10

D. 19

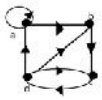
Q8. In the graph  the in-degree of vertex a is

A. deg-(a) = 3

B. deg-(a) = 4

C. deg-(a) = 5

D. deg-(a) = 6

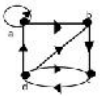
Q9. In the graph  the out-degree of vertex a is

A. deg+ (a) = 1

B. deg+ (a) = 2

C. deg+ (a) = 10

D. deg+ (a) = 12

Q10. In the graph  the sum of the in-degrees is

A. 7

B. 17

C. 10

D. 12

Q11. What does the degree of a vertex represent in an academic collaboration graph?

A. the number of collaborators v has

B. the set of coauthors

C. a person who has no coauthors

D. a person who has published with just one other person

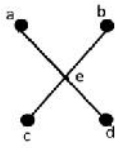
Q12. What does the neighborhood of a vertex represent in an academic collaboration graph?

A. the number of collaborators v has
B. the set of coauthors
C. a person who has no coauthors
D. a person who has published with just one other person

Q13. What does the isolated vertex represent in an academic collaboration graph?

A. the number of collaborators v has
B. the set of coauthors
C. a person who has no coauthors

Q14. a person who has published with just one other person



is a

A. Bipartite graph
B. Cycle
C. Wheel
D. $K_{3,4}$

Q15. How many vertices and how many edges do theKn graph have?

A. n(n - 1)/2
B. n(n - 1)/3
C. n(n - 1)/4
D. n(n - 1)/6

Q16.How many vertices and how many edges do theCn graphs have?

A. n(n - 1)
B. n
C. n(n - 2)
D. (n - 1)

Answer for Self Assessment

1.	C	2.	C	3.	B	4.	C	5.	D
6.	D	7.	C	8.	A	9.	A	10.	A
11.	A	12.	B	13.	C	14.	A	15.	A
16.	B								

Review Questions

Q1. Can a simple graph exist with 15 vertices each of degreefive?

Q2. Show that the sum, over the set of people at a party, ofthe number of people a person has shaken hands with, iseven. Assume that no one shakes his or her own hand.

Q3. What does the degree of a vertex represent in the acquaintanceshipgraph, where vertices represent all the peoplein the world?

Q4. What does the degree of a vertex in the Hollywood graphrepresent? What does the neighborhood of a vertex represent?What do the isolated and pendant vertices represent?

Q5. What do the in-degree and the out-degree of a vertex in adirected graph modeling a round-robin tournament represent?

Q6. What does the neighborhood a vertex in acquaintanceshipgraph represent?

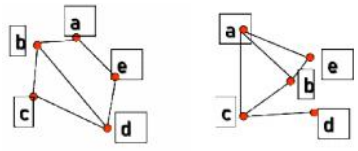
Q7. What do isolated and pendant verticesinacquaintanceshipgraphrepresent?

Q8. In one study it was estimated thatthe average degree of a vertex in acquaintanceshipgraph is 1000. Whatdoes this mean in terms of the model?

Q9. Draw these graphs.a) K_7 b) $K_{1,8}$ c) $K_{4,4}$d) C_7 e) W_7f) Q_4

Q10. For which values of n istheCn bipartite?

Q11. Are the two graph Isomorphic?



Further Readings:

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit08: Connectivity

CONTENTS

Objectives
Introduction
8.1 Connectivity
8.2 Cutpoints and Bridges
Summery
Self Assessment
Answer for Self Assessment
Review Questions
Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- understand what are path, cycles and connectivity in a graph.
- understand how to identify non-isomorphic graph with the help of cycles.

Introduction

8.1 Connectivity

A path in a multigraph G consists of an alternating sequence of vertices and edges of the form v_0, e_1, v_1, e_2, v_2, ..., e_{n-1}, v_{n-1}, e_n, v_n where each edge ei contains the vertices v_{i-1} and v_i (which appear on the sides of e_i in the sequence). The number n of edges is called the length of the path. When there is no ambiguity, we denote a path by its sequence of vertices (v_0, v_1,...,v_n). The path is said to be closed if $v_0 = v_n$. Otherwise, we say the path is from v_0, to v_n or between v_0 and v_n, or connects v_0 to v_n. A simple path is a path in which all vertices are distinct. (A path in which all edges are distinct will be called a trail.) A cycle is a closed path of length 3 or more in which all vertices are distinct except $v_0 = v_n$. A cycle of length k is called a k-cycle.

Many problems can be modelled with paths formed by travelling along the edges of graphs. For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model. Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on can be solved using models that involve paths in graphs.

Paths

Informally, a path is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph.As the path travels along its edges, it visits the vertices along this path, that is, the endpoints of these edges.

A formal definition of paths and related terminology is given in Definition 1.

DEFINITION 1 Let n be a nonnegative integer and G an undirected graph. A path of length n from u to v in G is a sequence of n edges $e_1,...,e_n$ of G for which there exists a sequence $x_0 = u, x_1,...,x_{n-1}, x_n = v$ of vertices such that e_i has, for i = 1,...,n, the end points x_{i-1} and x_i. When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, ..., x_n$, (because listing these vertices uniquely determines the path),the path is a circuit if it begins and ends at the same vertex, that is, if u = v, and has length greater than zero. The path or circuit is said to pass through the

vertices $x_1, x_2 \ldots, x_{n-1}$ or traverse the edges $e_1, e_2, \ldots, e_n$. A path or circuit is simple if it does not contain the same edge more than once.When it is not necessary to distinguish between multiple edges, we will denote a path

$e_1, e_2, \ldots, e_n$, where e_i is associated with $\{x_{i-1}, x_i\}$ for i = 1,2,..., n by its vertex sequence $x_0, x_1, \ldots, x_n$. This notation identifies a path only as far as which vertices it passes through.Consequently, it does not specify a unique path when there is more than one path that passes through this sequence of vertices, which will happen if and only if there are multiple edges between some successive vertices in the list. Note that a path of length zero consists of a single vertex.

Remark: There is considerable variation of terminology concerning the concepts defined in Definition 1. For instance, in some books, the term walk is used instead of path, where a walk is defined to be an alternating sequence of vertices and edges of a graph, $v_0 e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n$, where v_{i-1} and v_i are the endpoints of e_i for i = 1,2,...,n When this terminology is used, closed walk is used instead of circuit to indicate a walk that begins and ends at the same vertex, and trail is used to denote a walk that has no repeated edge (replacing the term simple path). When this terminology is used, the terminology path is often used for a trail with no repeated vertices, conflicting with the terminology in Definition 1. Because of this variation in terminology, you will need to make sure which set of definitions are used in a particular book or article when you read about traversing edges of a graph. The text [GrYe06] is a good reference for the alternative terminology described in this remark.

1. In the simple graph shown in Figure below, a, d, c, f, e is a simple path of length 4, because {a, d}, {d, c}, {c, f}, and {f, e} are all edges. However, d, e, c, a is not a path, because {e, c}is not an edge. Note that b, c, f, e, b is a circuit of length 4 because {b, c}, {c, f}, {f, e}, and {e, b} are edges, and this path begins and ends at b. The path a, b, e, d, a, b, which is of length 5, is not simple because it contains the edge {a, b} twice. ▲

Paths and circuits in directed graphs were introduced in Chapter 9. We now provide more general definitions.

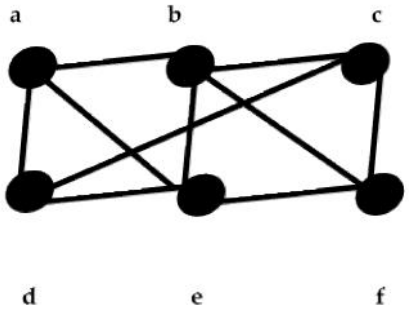


FIGURE 1 A Simple Graph.

DEFINITION 2 Let n is a nonnegative integer and G a directed graph.A path of length n from u to v in G is a sequence of edges $e_1, e_2, \ldots, e_n$ of G such that e1 is associated with (x_0, x_1), e_2 is associated with (x_1, x_2), and so on, with e_n associated with (x_{n-1}, x_n), where x_0 = u and x_n = v.When there are no multiple edges in the directed graph, this path is denoted by its vertex sequence $x_0, x_1, x_2, \ldots, x_n$.A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle. A path or circuit is called simple if it does not contain the same edge more than once.

Remark: TerminologyotherthanthatgiveninDefinition2isoftenusedfortheconceptsdefined there. In particular, the alternative terminology that uses walk, closed walk, trail, and path (describedin their marksfollowingDefinition1) may be used for directed graphs.

Note that the terminal vertex of an edge in a path is the initial vertex of the next edge in the path. When it is not necessary to distinguish between multiple edges, we will denote a path $e_1, e_2, \ldots, e_n$, where e_i is associated with (x_{i-1}, x_i) for i = 1, 2..., n, by its vertex sequence x_0, x_1, x_n.

The notation identifies a path only as far as which the vertices it passes through. There may be more than one path that passes through this sequence of vertices, which will happen if and only if there are multiple edges between two successive vertices in the list. Paths represent useful information in many graph models, as Examples 2–4 demonstrate.

Consider the graph G in figure below. Consider the following sequences: $\alpha = (P_4, P_1, P_2, P_5, P_1, P_2, P_3, P_6)$, $\beta = (P_4, P_1, P_5, P_2, P_6)$, $\gamma = (P_4, P_1, P_5, P_2, P_3, P_5, P_6)$, $\delta = (P_4, P_1, P_5, P_2, P_6)$. The sequence α is a path from P_4 to P_6; but it is not a trail since the edge $\{P_1, P_2\}$ is used twice. The sequence β is not a path since there is no edge $\{P_2, P_6\}$. The sequence γ is a trail since no edge is used twice; but it is not a simple path since the vertex P5 is used twice. The sequence δ is a simple path from P_4 to P_6; but it is not the shortest path (with respect to length) from P_4 to P_6. The shortest path from P_4 to P_6 is the simple path (P_4, P_5, P_6) which has length 2.

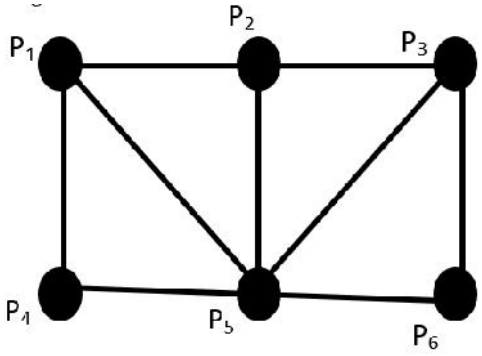


Figure 2

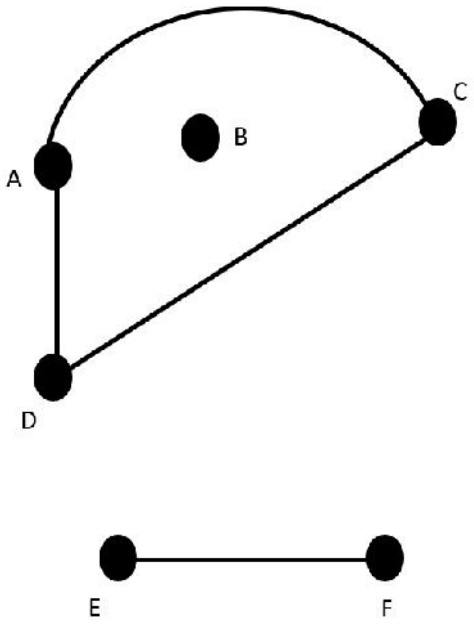


Figure 3

By eliminating unnecessary edges, it is not difficult to see that any path from a vertex u to a vertex v can be replaced by a simple path from u to v. We state this result formally.

Theorem: There is a path from a vertex u to a vertex v if and only if there exists a simple path from u to v.

Connectivity, Connected Components A graph G is connected if there is a path between any two of its vertices. The graph in Fig. 2 is connected, but the graph in Fig. 3 is not connected since, for example, there is no path between vertices D and E.

Suppose G is a graph. A connected subgraph H of G is called a connected component of G if H is not contained in any larger connected subgraph of G. It is intuitively clear that any graph G can be partitioned into its connected components. For example, the graph G in Fig. 3 has three connected components, the subgraphs induced by the vertex sets {A, C, D}, {E, F}, and {B}.

The vertex B in Fig. 3 is called an isolated vertex since B does not belong to any edge or, in other words, deg (B) = 0. Therefore, as noted, B itself forms a connected component of the graph.

Remark: Formally speaking, assuming any vertex u is connected to itself, the relation "u is connected to v" is an equivalence relation on the vertex set of a graph G and the equivalence classes of the relation form the connected components of G.

Distance and Diameter

Consider a connected graph G. The distance between vertices u and v in G, written d (u, v), is the length of the shortest path between u and v. The diameter of G, written diam (G), is the maximum distance between any two points in G. For example, in Fig. 4, d (A, F) = 2 and diam (G) = 3, whereas in Fig. 5, d (A, F) = 3 and diam(G) = 4.

8.2 Cutpoints and Bridges

Let G be a connected graph. A vertex v in G is called a cutpoint if G − v is disconnected. (Recall that G – v is the graph obtained from G by deleting v and all edges containing v.) An edge e of G is called a bridge if G – e is disconnected. (Recall that G − e is the graph obtained from G by simply deleting the edge e). In Fig. 4, the vertex D is a cutpoint and there are no bridges. In Fig. 5, the edge = {D, F} is a bridge. (Its endpoints D and F are necessarily Cutpoints.)

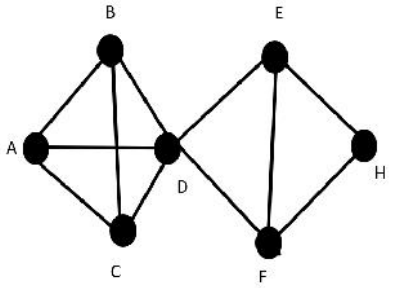


Figure 4

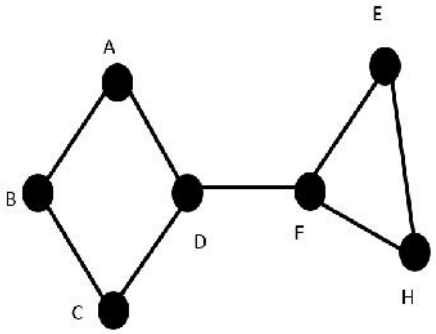
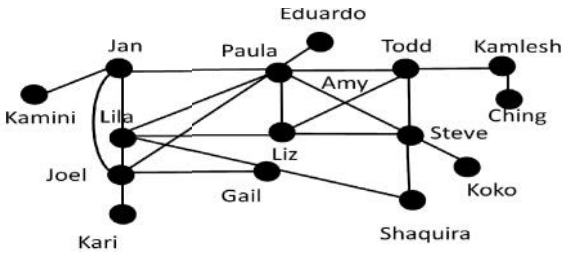


Figure 5

2. Paths in Acquaintanceship Graphs In an acquaintanceship graph there is a path between two people if there is a chain of people linking these people, where two people adjacent in the chain know one another. For example, in Figure below, there is a chain of six people linking Kamini and Ching. Many social scientists have conjectured that almost every pair of people in the world is linked by a small chain of people, perhaps containing just five or fewer people. This would mean that almost every pair of vertices in the acquaintanceship graph containing all people in the world is linked by a path of length not exceeding four. The play Six Degrees of Separation by John Guare is based on this notion. ▲



3. Paths in Collaboration Graphs In a collaboration graph, two people a and b are connected by a path when there is a sequence of people starting with a and ending with b such that the endpoints of each edge in the path are people who have collaborated. We will consider two particular collaboration graphs here. First, in the academic collaboration graph of people who have written papers in mathematics, the Erdo˝s number of a person m is the length of the shortest path between m and the extremely prolific mathematician Paul Erdo˝s (who died in 1996). That is, the Erdo˝s number of a mathematician is the length of the shortest chain of mathematicians that begins with Paul Erdo˝s and ends with this mathematician, where each adjacent pair of mathematicians have written a joint paper. The number of mathematicians with each Erdo˝s number as of early 2006, according to the Erdo˝s Number Project, is shown in Table 1. In the Hollywood graph (see Example 3 in Section 10.1) two actors a and b are linked when there is a chain of actors linking a and b, where every two actors adjacent in the chain have acted in the same movie. In the Hollywood graph, the Bacon number of an actor c is defined to be the length of the shortest path connecting c and the well-known actor Kevin Bacon. As new movies are made, including new ones with Kevin Bacon, the Bacon number of actors can change. In Table 2 we show the number of actors with each Bacon number as of early 2011 using data from the Oracle of Bacon website. The origins of the Bacon number of an actor dates back to the early 1990s, when Kevin Bacon remarked that he had worked with everyone in Hollywood or someone who worked with them. This led some people to invent a party

TABLE 1 The Number of Mathematicians with a Given Erdo˝s Number (as of early 2006).

Erdo˝s Number	Number of People
0	1
1	504
2	6593
3	33,605
4	83,642
5	87,760
6	40,014
7	11,591
8	3,146
9	819
10	244
11	68
12	23
13	5

TABLE 2 The Number of Actors with a Given Bacon Number (as of early 2011).

Bacon Number	Number of People
0	1
1	2,367
2	242,407
3	785,389
4	200,602
5	14,048
6	1,277
7	114
8	16

game where participants where challenged to find a sequence of movies leading from each actor named to Kevin Bacon. We can find a number similar to a Bacon number using any actor as the centre of the acting universe. ▲

Connectedness in Undirected Graphs

When does a computer network have the property that every pair of computers can share information, if messages can be sent through one or more intermediate computers? When a graph is used to represent this computer network, where vertices represent the computers and edges represent the communication links, this question becomes: When is there always a path between two vertices in the graph?

DEFINITION 3 An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph. An undirected graph that is not connected is called disconnected. We say that we disconnect a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.

Thus, any two computers in the network can communicateifandonlyifthegraphofthisnetwork is connected.

4. The graph G1 in Figure below is connected; because for every pair of distinct vertices there is a path between them (the reader should verify this). However, the graph G2in Figure below is not connected. For instance, there is no path in G_2 between vertices a and d. ▲

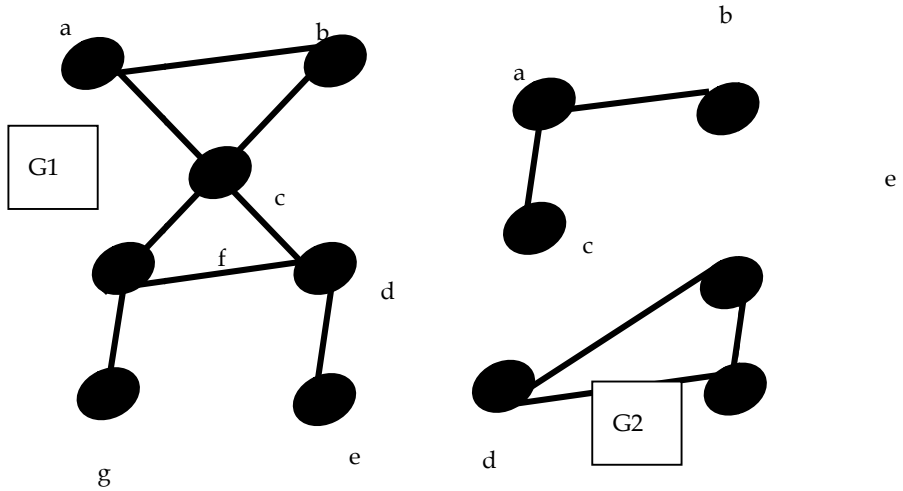
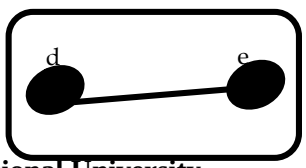


Figure 6



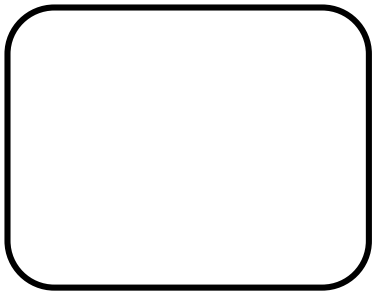


Figure 8

FIGURE 7 The Graph H and its connected components H_1, H_2, And H_3

THEOREM1. Thereisasimplepathbetweeneverypairofdistinct verticesofaconnectedundirectedgraph.

Proof: Let u and v be two distinct vertices of the connected undirected graph G = (V, E). Because Gisconnected,thereisatleastonepathbetweenuandv.Let $x_0,x_1,...,x_n$,where x_0=u andx_n = v,bethe vertexes quinceofapathofleastlength.Thispathofleastlengthissimple,to see this, suppose it is not simple.Then $x_i = x_j$ for some i and j with $0 \le i<j$.This means that there is a path from u to v of shorter length with vertex sequence $x_0,x_1,...,x_{i-1},x_j,...,x_n$ obtained by deleting the edges corresponding to the vertex sequence $x_i,...,x_{j-1}$.

CONNECTEDCOMPONENTS A connected component of a graph G is a connected sub graph of G that is not a proper sub graph of another connected sub graphof G.Thatis,aconnected component of a graph G is a maximal connected subgraph of G.A graph G that is not connected has two or more connected components that are disjoint and have G as their union.

5. What are the connected components of the graph H shown in Figure 3?

Solution: The graphH istheunionofthreedisjointconnectedsubgraphsH_1, H_2, andH_3, shown in Figure 7. These three subgraphs are the connected components of H. ▲

6. ConnectedComponentsofCallGraphs Twoverticesx andy areinthesamecomponentofa telephone call graph when there is a sequence of telephone calls beginning at x and ending at y. When a call graph for telephone calls made during a particular day in the AT&T network was analyzed, this graph was found to have 53,767,087 vertices, more than 170 million edges, and more than 3.7 million connected components. Most of these components were small; approximatelythree-fourthisconsistedoftwoverticesrepresentingpairs of telephone numbers that called only each other. This graphhasonehugeconnectedcomponent with 44,989,297 vertices comprising more than 80% of the total. Furthermore, every vertex in this component can be linked to any other vertex by a chain of no more than 20 calls.

How Connected is a Graph?

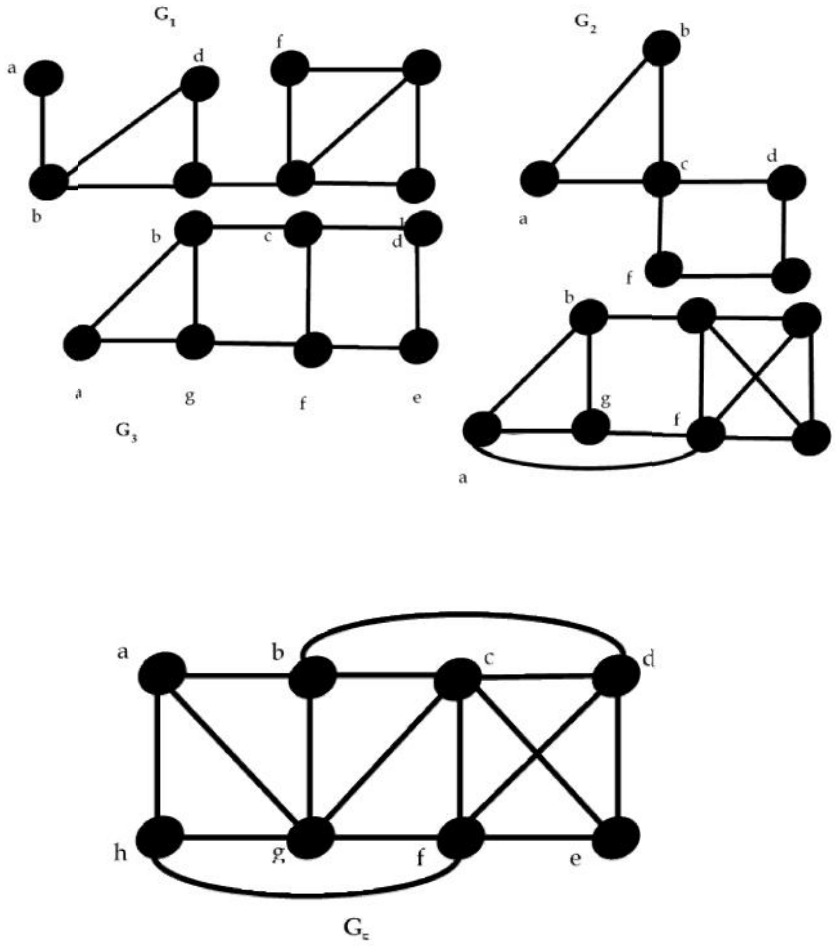
Suppose that a graph represents a computer network. Knowing that this graph is connected tells us that any two computers on the network can communicate. However, we would also like to understand how reliable this network is. For instance, will it still be possible for all computers to communicate after a router or a communications link fails? To answer this and similar questions, we now develop some new concepts.

Sometimes the removal from a graph of a vertex and all incident edges produces a subgraph with more connected components. Such verticesare called cutvertices(or articulationpoints). The removal of a cut vertex from a connected graph produces a subgraph that is not connected. Analogously, an edge whose removal produces a graph with more connected components than in the original graph is called a cut edge or bridge. Note that in a graph representing a computer network, acutvertexandacutedgerepresentanessentialrouterandanessentiallinkthatcannot fail for all computers to be able to communicate.

7. Find the cut vertices and cut edges in the graph G1 shown in Figure below.

Solution: The cut vertices of G_1 are b, c, and e. The removal of one of these vertices (and its adjacent edges) disconnects the graph. The cut edges are {a, b}and{c,e}. Removing either one of these edges disconnects G_1.

VERTEX CONNECTIVITY: Not all graphs have cut vertices. For example, the complete graph K_n, where $n \geq 3$, has no cut vertices. When you remove a vertex from K_n and all edges incident to it, the resulting subgraph is the complete graph K_{n-1}, a connected graph. Connected graphs without cut vertices are called no separablegraphs, and can be thought of as more connected than those with a cut vertex. We can extend this notion by defining a more granulated measure of graph connectivity based on the minimum number of vertices that can be removed to disconnect a graph.



A subset Vof the vertex set V of G = (V, E) is a vertex cut, orseparating set, ifG−Vis disconnected. For instance, in the graph in Figure 1, the set {b, c, e}is a vertex cut with three vertices, as the reader should verify. We leave it to the reader to show that every connected graph, except a complete graph, has a vertex cut. We define the vertex connectivity of a non complete graph G, denoted by $\kappa(G)$, as the minimum number of vertices in a vertex cut.

When G is a complete graph, it has no vertex cuts, because removing any subset of its vertices and all incident edges still leaves a complete graph. Consequently, we cannot define $\kappa(G)$ as the minimum number of vertices in a vertex cut when G is complete. Instead, weset$\kappa(K_n) = n-1$, the number of vertices needed to be removed to produce a graph with a single vertex. Consequently, for every graph G, $\kappa(G)$ is minimum number of vertices that can be removed from G to either disconnect G or produce a graph with a single vertex. We have $0 \leq \kappa(G) \leq n-1$ ifG has n vertices, $\kappa(G) = 0$ if and only if G is disconnected or G = K1, and $\kappa(G) = n-1$ if and only if G is complete. The larger $\kappa(G)$ is, the more connected we consider G to be. Disconnected graphs and K1 have $\kappa(G) = 0$, connected graphs with cut vertices and K2 have $\kappa(G) = 1$, graphs without cut vertices that can be disconnected by removing two vertices and K3 have $\kappa(G) = 2$, and so on. We say that a graph is k-connected (or k-vertex-connected), if $\kappa(G) \geq k$. A graph G is 1connected if it is connected and not a

graph containing a single vertex; a graph is 2-connected, or biconnected, if it is non-separable and has at least three vertices. Note that if G is a k-connected graph, then G is a j-connected graph for all j with $0 \leq j \leq k$.

8. Find the vertex connectivity for each of the graphs in Figure above.

Solution: Each of the five graphs in Figure above is connected and has more than vertex, so each of these graphs has positive vertex connectivity. Because G_1 is a connected graph with a cut vertex, as shown in Example 7, we know that $\kappa(G_1) = 1$. Similarly, $\kappa(G_2) = 1$, because c is a cut vertex of G_2. The reader should verify that G_3 has no cut vertices. But that {b, g} is a vertex cut. Hence, $\kappa(G_3) = 2$. Similarly, because G_4 has a vertex cut of size two, {c, f}, but no cut vertices. It follows that $\kappa(G_4) = 2$. The reader can verify that G_5 has no vertex cut of size two, but {b, c, f} is a vertex cut of G_5. Hence, $\kappa(G_5) = 3$.

EDGE CONNECTIVITY We can also measure the connectivity of a connected graph G = (V, E) in terms of the minimum number of edges that we can remove to disconnect it. If a graph has a cut edge, then we need only remove it to disconnect G. IfG does not have a cut edge, we look for the smallest set of edges that can be removed to disconnect it. A set of edges E′ is called an edge cut of G if the subgraph $G-E'$ is disconnected. The edge connectivityλ is the lowercase Greek letter lambda. Of a graph G, denoted by λ (G), is the minimum number of edges in an edge cut of G. This defines λ (G) for all connected graphs with more than one vertex because it is always possible to disconnect such a graph by removing all edges incident to one of its vertices Note that λ(G) = 0 ifG is not connected. We also specify that λ(G) = 0 ifG is a graph consisting of a single vertex. It follows that if G is a graph with n vertices, then $0 \leq \lambda(G) \leq n-1$. Reader can show that $\lambda(G) = n-1$ where G is a graph with n vertices if and only if $G = K_n$, which is equivalent to the statement that $\lambda(G) \leq n-2$ when G is not a complete graph.

9. Find the edge connectivity of each of the graphs in Figure above.

Solution: Each of the five graphs in Figure 4 is connected and has more than one vertex, so we know that all of them have positive edge connectivity. As we saw in Example 7, G_1 has a cut edge, so $\lambda(G_1) = 1$.

The graph G_2 has no cut edges, as the reader should verify, but the removal of the two edges {a, b} and {a, c} disconnects it. Hence, $\lambda(G_2) = 2$. Similarly, $\lambda(G_3) = 2$, because G_3 has no cut edges, but the removal of the two edges {b, c} and {f, g} disconnects it. The reader should verify that the removal of no two edges disconnects G_4, but the removal of the three edges {b, c}, {a, f}, and {f, g} disconnects it. Hence, $\lambda(G_4) = 3$. Finally, the reader should verify that $\lambda(G_5) = 3$, because the removal of any two of its edges does not disconnect it, but the removal of {a, b}, {a, g}, and {a, h} does.

AN INEQUALITY FOR VERTEX CONNECTIVITY AND EDGE CONNECTIVITY When G = (V, E) is a noncomplete connected graph with at least three vertices, the minimum degree of a vertex of G is an upper bound for both the vertex connectivity of G and the edge connectivity of G. That is, $\kappa(G) \leq \min_{v \in V} \deg(v)$ and $\lambda(G) \leq \min_{v \in V} \deg(v)$. To see this, observe that deleting all the neighbours of a fixed vertex of minimum degree disconnects G, and deleting all the edges that have a fixed vertex of minimum degree as an endpoint disconnects G. Reader can show that $\kappa(G) \leq \lambda(G)$ when G is a connected noncomplete graph. Note also that $\kappa(K_n) = \lambda(K_n) = \min_{v \in V} \deg(v) = n-1$ when n is a positive integer and that $\kappa(G) = \lambda(G) = 0$ when G is a disconnected graph. Putting these facts together, establishes that for all graphs G,

$\kappa(G) \leq \lambda(G) \leq \min_{v \in V} \deg(v)$.

APPLICATIONS OF VERTEX AND EDGE CONNECTIVITY: Graph connectivity plays an important role in many problems involving the reliability of networks. For instance, as we mentioned in our introduction of cut vertices and cut edges, we can model a data network using vertices to represent routers and edges to represent links between them. The vertex connectivity of the resulting graph equals the minimum number of routers that disconnect the network when they are out of service. If fewerroutersaredown, data transmission between every pair of routers is still possible. The edge connectivity represents the minimum number of fiber optic links that can be down to disconnect the network. If fewer links are down, it will still be possible for data to be transmitted between every pair of routers. We can model a highway network, using vertices to represent highway intersections and edges to represent sections of roads running between inter sections. The vertexconnectivityofthe resultinggraphrepresentstheminimumnumberofintersectionsthatcanbeclosedataparticular time that

makes it impossible to travel between every two intersections. If fewer intersections are closed, travel between every pair of intersections is still possible. The edge connectivity represents the minimum number of roads that can be closed to disconnect the highway network. If fewer highways are closed, it will still be possible to travel between any two intersections. Clearly, it would be useful for the highway department to take this information into account when planning road repairs.

Connectedness in Directed Graphs

There are two notions of connectedness in directed graphs, depending on whether the directions of the edges are considered.

DEFINITION 4 A directed graph is strongly connected if there is a path from a to b and from b to a whenever a and b are vertices in the graph.

For a directed graph to be strongly connected there must be a sequence of directed edges from any vertex in the graph to any other vertex. A directed graph can fail to be strongly connected but still be in "one piece." Definition 5 makes this notion precise.

DEFINITION 5 A directed graph is weakly connected if there is a path between every two vertices in the underlying undirected graph.

That is, a directed graph is weakly connected if and only if there is always a path between two vertices when the directions of the edges are disregarded. Clearly, any strongly connected directed graph is also weakly connected.

10. Are the directed graphs G and H shown in Figure 8 strongly connected? Are they weakly connected?

Solution: G is strongly connected because there is a path between any two vertices in this directed graph (the reader should verify this). Hence, G is also weakly connected. The graph H is not strongly connected. There is no directed path from a to b in this graph. However, H is weakly connected, because there is a path between any two vertices in the underlying undirected graph of H (the reader should verify this). ▲

STRONGCOMPONENTSOFADIRECTEDGRAPH

ThesubgraphsofadirectedgraphG that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the strongly connected components or strong components of G. Note that if a and b are two vertices in a directed graph, their strong components are either the same or disjoint.

11. The graph H in Figure 8 has three strongly connected components, consisting of the vertex a; the vertex e; and the subgraph consisting of the vertices b, c, and d and edges (b, c), (c, d), and (d, b).

12. The Strongly Connected Components of the Web Graph the Web graph introduced in Example 5 of Section 10.1 represents Web pages with vertices and links with directed edges. A snapshot of the Web in 1999 produced a Web graph with over 200 million vertices and over 1.5 billion edges (numbers that have now grown considerably). (See [Br00] for details.) The underlying undirected graph of this Web graph is not connected, but it has a connected component that includes approximately 90% of the vertices in the graph. The subgraph of the This implies that more than 40 TB of computer memory would have been needed to represent its adjacency matrix. Original directed graph corresponding to this connected component of the underlying undirected graph (that is, with the same vertices and all directed edges connecting vertices in this graph) has one very large strongly connected component and many small ones. The former is called the giant strongly connected component (GSCC) of the directed graph. A Web page in this component can be reached following links starting at any other page in this component. The GSCC in the Web graph produced by this study was found to have over 53 million vertices. The remaining vertices in the large connected component of the undirected graph represent three different types of Web pages: pages that can be reached from a page in the GSCC, but do not link back to these pages following a series of links; pages that link back to pages in the

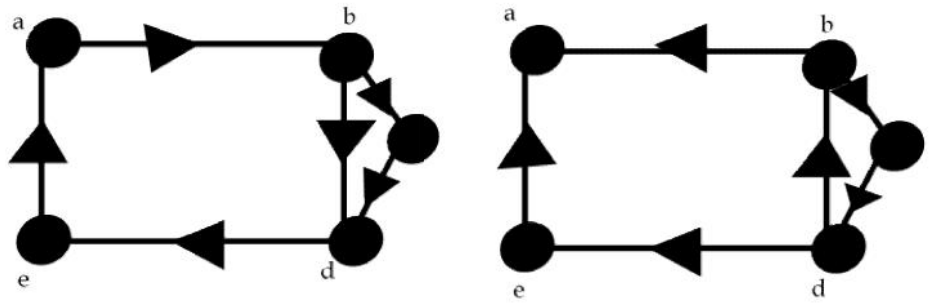


FIGURE 8the directed graph g and h

GSCC following a series of links, but cannot be reached by following links on pages in the GSCC; and pages that cannot reach pages in the GSCC and cannot be reached from pages in the GSCC following a series of links. In this study, each of these three other sets was found to have approximately 44 million vertices. (It is rather surprising that these three sets are close to the same size.) ▲

Paths and Isomorphism

There are several ways that paths and circuits can help determine whether two graphs are isomorphic. For example, the existence of a simple circuit of a particular length is a useful invariant that can be used to show that two graphs are not isomorphic. In addition, paths can be used to construct mappings that may be isomorphism. As we mentioned, a useful isomorphic invariant for simple graphs is the existence of a simple circuit of length k, where k is a positive integer greater than 2. Example 13 illustrates how this invariant can be used to show that two graphs are not isomorphic.

13. Determine whether the graphs G and H shown in Figure below are isomorphic.

Solution: Both G and H have six vertices and eight edges. Each has four vertices of degree three, and two vertices of degree two. So, the three invariants—number of vertices, number of edges, and degrees of vertices—all agree for the two graphs. However, H has a simple circuit of length three, namely, v1, v2, v6, v1, whereas G has no simple circuit of length three, as can be determined by inspection (all simple circuits in G have length at least four). Because the existence of a simple circuit of length three is an isomorphic invariant, G and H are not isomorphic. ▲

We have shown how the existence of a type of path, namely, a simple circuit of a particular length, can be used to show that two graphs are not isomorphic. We can also use paths to find mappings that are potential isomorphism.

14. Determine whether the graphs G and H shown in Figure below are isomorphic.

Solution: Both G and H have five vertices and six edges, both have two vertices of degree three and three vertices of degree two, and both have a simple circuit of length three, a simple circuit of length four, and a simple circuit of length five. Because all these isomorphic invariants agree, G and H may be isomorphic.

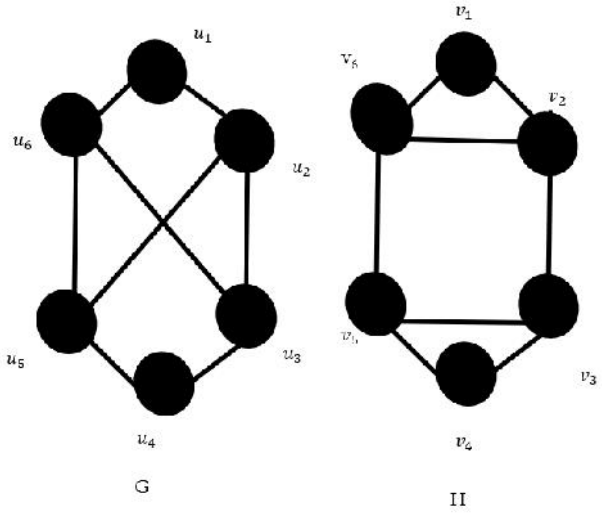


FIGURE 9

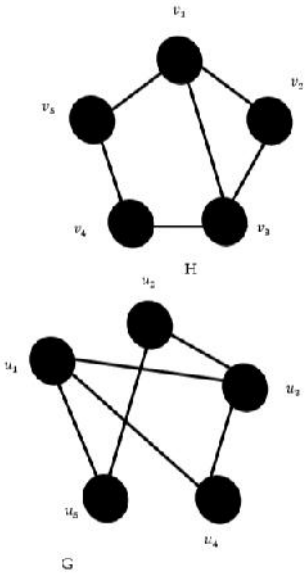


FIGURE 10

To find a possible isomorphism, we can follow paths that go through all vertices so that the corresponding vertices in the two graphs have the same degree. For example, the paths u_1, u_4, u_3, u_2, u_5 in G and v_3, v_2, v_1, v_5, v_4 in H both go through every vertex in the graph; start at a vertex of degree three; go through vertices of degrees two, three, and two, respectively; and end at a vertex of degree two. By following these paths through the graphs, we define the mapping f with $f(u_1) = v_3$, $f(u_4) = v_2$, $f(u_3) = v_1$, $f(u_2) = v_5$, and $f(u_5) = v_4$. The reader can show that f is an isomorphism, so G and H are isomorphic, either by showing that f preserves edges or by showing that with the appropriate orderings of vertices the adjacency matrices of G and H are the same.

Counting Paths between Vertices

The number of paths between two vertices in a graph can be determined using its adjacency matrix.

THEOREM 2 Let G be a graph with adjacency matrix A with respect to the ordering $v_1, v_2, \ldots, v_n$ of the vertices of the graph (with directed or undirected edges, with multiple edges and loops allowed).The number of different paths of length r from v_i to v_j, where r is a positive integer, equals the (i,j)th entry of A^r.

Proof: The theorem will be proved using mathematical induction. Let G be a graph with adjacency matrix A (assuming an ordering v_1, v_2, v_n of the vertices of G). The number of paths from vi to vj of

length 1 is the (i, j) th entry of A, because this entry is the number of edges from v_i to v_j. Assume that the (i, j) th entry of A^r is the number of different paths of length r from v_i to v_j. This is the inductive hypothesis. Because $A^{r+1} = A^rA$, the (i, j) th entry of A^{r+1} equals $b_{i1a1j} + b_{i2a2j} + \cdots + binanj$, where bik is the (i, k) th entry of Ar. By the inductive hypothesis, bik is the number of paths of length r from vi to v_k. A path of length r +1 from v_i to v_j is made up of a path of length r from vi to some intermediate vertex v_k, and an edge from v_k to vj. By the product rule for counting, the number of such paths is the product of the number of paths of length r from v_i to v_k, namely, bik, and the number of edges from v_k to v_j, namely, ak_j. When these products are added for all possible intermediate vertices vk, the desired result follows by the sum rule for counting.

15. How many paths of length four are there from a to d in the simple graph G in Figure 11?

Solution: The adjacency matrix of G (ordering the vertices as a, b, c, d) is

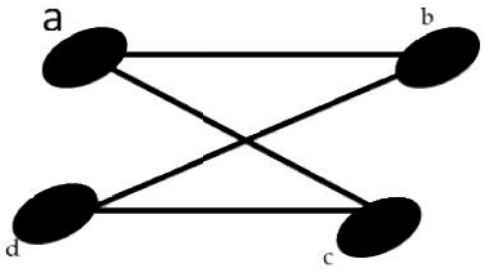


FIGURE 11

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Hence, the number of paths of length four from a to d is the (1, 4) th entry of A^4. Because

$$\begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

There are exactly eight paths of length four from a to d. By inspection of the graph, we see that a,b,a,b,d; a,b,a,c,d; a,b,d,b,d; a,b,d,c,d; a,c,a,b,d; a,c,a,c,d; a,c,d,b,d; and a,c,d,c,d are the eight paths of length four from a to d. ▲

Theorem 2 can be used to find the length of the shortest path between two vertices of a graph, and it can also be used to determine whether a graph is connected.

Summery

- We have learned what are path, cycles and connectivity in a graph.
- We have learned how to identify non-isomorphic graph with the help of cycles.

Self Assessment

1. In a tree between every pair of vertices there is?

   A. Exactly one path
   B. A self loop
   C. Two circuits
   D. n number of paths

2. If the origin and terminus of a walk are same, the walk is known as...?

   A. Open
   B. Closed
   C. Path
   D. None of these

3. Which of the following is not a type of graph?

   A. Euler
   B. Hamiltonian
   C. Tree
   D. Path

4. Polyhedral is....?

   A. A simple connected graph
   B. A plane graph
   C. A graph in which the degree of every vertex and every face is atleast3
   D. All of above

5. Which of the following statement is false?

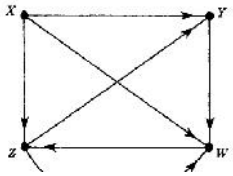
   A. G is connected and is circuitless
   B. G is connected and has n edges
   C. G is minimally connected graph
   D. G is circuitless and n-1 edges

6. The number of paths of length n between two different vertices in K_4 if n is 5 are

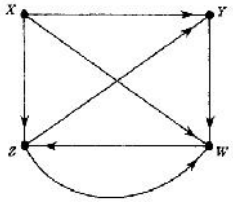
   A. 71
   B. 61
   C. 23
   D. 41

7. The number of paths between c and d in the graph in Figure 2 of length (a) 2

   A. 7
   B. 4
   C. 1
   D. 0

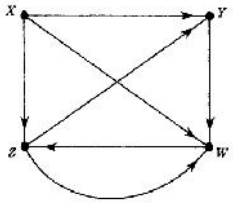
8. Let *G* be the directed graph 

hen total number cycles in *G* are/is
   A. There is only one cycle in *G*, which is *(Y, W, Z, Y)*.
   B. There 2 one cycles
   C. There 3 one cycles
   D. There 4 one cycles

9. Let *G* be the directed graph 

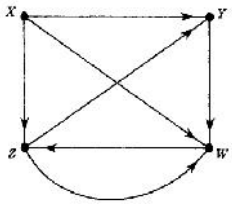
hen all simple paths from *X* to *Z*are

   A. There are three simple paths from *X* to *Z*, which are *(X, Z)*, *(X, W, Z)*, and *(X, Y, W, Z)*.
   B. There are four simple paths from *X* to *Z*, which are *(X, Z)*, *(X, W, Z)*, and *(X, Y, W, Z)*, *(X, Y, W)*.
   C. There are two simple paths from *X* to *Z*, which are *(X, Z)*, and *(X, Y, W, Z)*.
   D. There is one simple paths from *X* to *Z*, which are *(X, Z)*.

10. Let *G* be the directed graph 

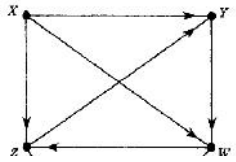
All simple paths from *Y* to *Z*are

   A. There is only one simple path from *Y* to *Z*, which is (*Y, W, Z*).
   B. There are only two simple paths from *Y* to *Z*, which is (*Y, W, Z*) and (*Y, X, W, Z*)
   C. There are only three simple paths from *Y* to *Z*, which is (*Y, W, Z*), (*Y, X, Z*) and (*Y, X, W, Z*)
   D. There are only four simple paths from *Y* to *Z*, which is (*Y, W, Z*) , (*Y, X, W*) , (*Y, X, Y*) and (*Y, X, W, Z*)

11. Let *G* be the directed graph in following Fig 

the in degree(X)

   A. 0
   B. 1
   C. 2
   D. 3

12. Let G be the directed graph in following Fig. 

he source/s is/are
    A. X
    B. Y
    C. W
    D. Z

13. For which of the following does there exist a simple graph G = (V, E) satisfying the specified conditions?

    A. It has 3 components 20 vertices and 16 edges.
    B. It has 6 vertices, 11 edges, and more than one component.
    C. It is connected and has 10 edges 5 vertices and fewer than 6 cycles.
    D. It has 7 vertices, 10 edges, and more than two components.

14. Let G be the directed graph with vertex set V (G) = (a, b, c, d, e, f, g} and edge set: E (G) = {(a, a), (b, e), (a, e), (e, b), (g, c), (a, e), (d, f), (d, b), (g, g)}. Then loops are
    A. (g, g)
    B. (a, a),(b, e), (e, b)
    C. (a, a)
    D. (a, a) and (g, g)

15. For which of the following does there exist a graph G = (V, E, φ) satisfying the specifiedconditions?

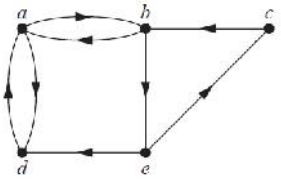
    A. A tree with 9 vertices and the sum of the degrees of all the vertices is 18.
    B. A graph with 5 components 12 vertices and 7 edges.
    C. A connected graph with 12 edges 5 vertices and fewer than 8 cycles
    D. A graph with 9 vertices, 9 edges, and no cycles.

Answer for Self Assessment

1.	A	2.	B	3.	D	4.	C	5.	B
6.	D	7.	C	8.	A	9.	A	10.	A
11.	A	12.	B	13.	D	14.	A	15.	B

Review Questions

1. Does each of these lists of vertices form a path in the following graph? Which paths are simple?



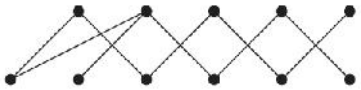
Which are circuits? What are the lengths of those that are paths?

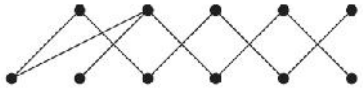
a) a, b, e, c, b

b) a, d, a, d, a

c) a, d, b, e, a

d) a, b, e, c, b, d, a

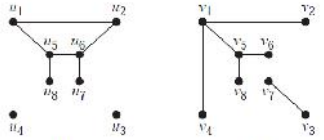
2. Determine whether the given graph is connected 



3. How many connected components does each of the graph have? For each graph find each of its connected components.

4. What do the connected components of a collaboration graph represent?

5. Use paths either to show that these graphs are not isomorphic or to find an isomorphism between these graphs.



6. Find the number of paths between c and d in the graph in Figure 1 of lengtha) 2. b) 3. c) 4. d) 5. e) 6. f) 7.

7. Show that every connected graph with n vertices has at least $n - 1$ edges.

8. Show that in every simple graph there is a path from every vertex of odd degree to some other vertex of odd degree.

9. Show that a vertex c in the connected simple graph G is a cut vertex if and only if there are vertices u and v, both different from c, such that every path between u and v passes through c.

10. Show that an edge in a simple graph is a cut edge if and only if this edge is not part of any simple circuit in the graph.

Further Reading

Rosen, Kenneth H. "Discrete Mathematics and Its Application."

Rosen, Kenneth H. and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.

Lipschutz, Seymour and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 09: Euler and Hamilton Paths

CONTENTS

Objective

Introduction

9.1 Euler Paths and Circuits

9.2 Necessary and Sufficient Conditions For Euler Circuits And Paths

9.3 Applications of Euler Paths and Circuits

9.4 Method for Finding Euler Circuit

9.5 Hamilton Paths and Circuits

9.6 Conditions for the Existence of Hamilton Circuits

Summary

Self Assessment

Answers for Self Assessment

Review Questions

Further Readings

Objective

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- Understand what are Euler path in an undirected graph
- Understand what are Euler path in a directed and undirected graph.
- Understand what are Hamilton path and graphs in a directed and undirected graph.

Introduction

Can we travel along the edges of a graph starting at a vertex and returning to it by traversing each edge of the graph exactly once? Similarly, can we travel along the edges of a graph starting at a vertex and returning to it while visiting each vertex of the graph exactly once? Although these questions seem to be similar, the first question, which asks whether a graph has a Euler circuit, can be easily answered simply by examining the degrees of the vertices of the graph, while the second question, which asks whether a graph has a Hamilton circuit, is quite difficult to solve for most graphs. In this section we will study these questions and discuss the difficulty of solving them. Although both questions have many practical applications in many different areas, both arose in old puzzles. We will learn about these old puzzles as well as modern practical applications.

9.1 Euler Paths and Circuits

The town of Konigsberg, Prussia (now called Kaliningrad and part of the Russian republic), was divided into four sections by the branches of the Pregel River. These four sections included the two regions on the banks of the Pregel, Kneiphof Island, and the region between the two branchesofthePregel. In the eighteenthcenturysevenbridgesconnectedtheseregions. Figure1 depicts these regions and bridges.

The townspeople took long walks through town on Sundays.They wondered whether it was possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice, and return to the starting point. The Swiss mathematician Leonhard Euler solved this problem. His solution, published in 1736, may be the first use of graph theory. Euler studied this

problem using the multigraph obtained when the four regions are represented by vertices and the bridges by edges. This multigraph is shown in Figure 2

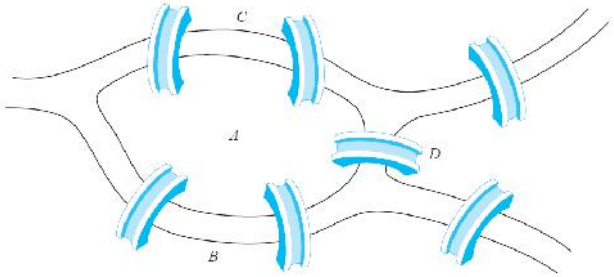


Figure 1: The Seven Bridges of Konigsberg.(Courtesy: Rosen, Kenneth H. "Discrete Mathematics and Its Applications.")

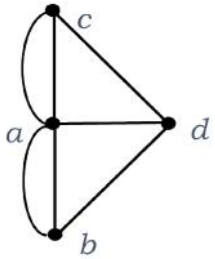


Figure 2: Multigraph model of the town of Konigsberg

The problem of traveling across every bridge without crossing any bridge more than once can be rephrased in terms of this model. The question becomes: Is there a simple circuit in this multigraph that contains every edge?

Definition 1 A Euler circuit in a graph G is a simple circuit containing every edge of G. AnEuler path in G is a simple path containing every edge of G.

Examples 1 and 2 illustrate the concept of Euler circuits and paths.

Example 1: Which of the undirected graphs in Figure 3 have a Euler circuit? Of those that do not, which have a Euler path?

Solution: The graph G_1 has a Euler circuit, for example, a, e, c, d, e, b, a. Neither of the graphs G_2 or G_3 has a Euler circuit (the reader should verify this). However, G_3 has a Euler path, namely, a, c, d, e, b, d, a, b. G_2 does not have a Euler path (as the reader should verify).

Example 2:Which of the directed graphs in Figure 4 have a Euler circuit? Of those that do not, which have a Euler path?

Solution: The graph H2 has a Euler circuit, for example, a, g, c, b, g, e, d, f, a. Neither H1 nor H3 has a Euler circuit (as the reader should verify). H3 has a Euler path, namely, c, a, b, c, d, b, butH1 does not (as the reader should verify).

Definition: A path in a graph G is called an Euler Path if it includes everyedge exactly once.

Definition: A circuit in a graph G is called an Euler Circuit if it includes every edge exactly once. Thus, an Euler circuit (Eulerian trail) for a graph G is a sequence of adjacent vertices and edges in G that starts and ends at the same vertex, uses every vertex of G at least once, and uses every edge of G exactly once.

A graph is called Eulerian graph if there exists a Euler circuit for that graph.

Theorem. If a graph has an Euler circuit, then every vertex of the graph haseven degree.

A finite connected graph G has an Euler circuit if and only if every vertex of G has even degree.

Thus finite connected graph is Eulerian if and only if each vertex has even degree.

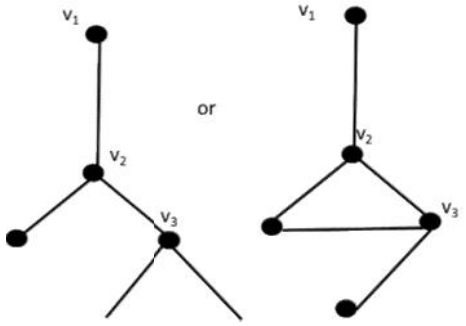
Proof: Let G be a graph which has an Euler circuit. Let v be a vertex of G. We shall show that degree of v is even. By definition, Euler circuit contains every edge of graph G. Therefore the Euler circuit contains all edges incident on v.

We start a journey beginning in the middle of one of the edges adjacent to the start of Euler circuit and continue around the Euler circuit to end in the middle of the starting edge. Since Euler circuit uses every edge exactly once, the edges incident on v occur in entry / exist pair and hence the degree of v is a multiple of 2. Therefore the degree of v is even. This completes the proof of the theorem.

We know that contrapositive of a conditional statement is logically equivalentto statement. Thus the above theorem is equivalent to:

Theorem5. If a graph G has more than two vertices of odd degree, then there can be no Euler path in G.

Proof : Let v1, v2 and v3 be vertices of odd degree. Since each of these vertices had odd degree, any possible Euler path must leave (arrive at) each of v1, v2, v3 with no way to return (or leave). One vertex of these three vertices may be the beginning of Euler path and another the end but this leaves the third vertex at one end of an untravelled edge. Thus there is no Euler path.



(Graphs having more than two vertices of odd degree).

Theorem:2. If a vertex of a graph is not of even degree, then it does not have an Euler circuit.

or

"If some vertex of a graph has odd degree, then that graph does not have an Euler circuit".

9.2 Necessary and Sufficient Conditions For Euler Circuits And Paths

There are simple criteria for determining whether a multigraph has a Euler circuit or a Euler path. Euler discovered them when he solved the famous Konigsberg bridge problem. We will assume that all graphs discussed in this section have a finite number of vertices and edges. What can we say if a connected multigraph has a Euler circuit? What we can show is that every vertex must have even degree. To do this, first note that a Euler circuit begins with a vertex a and continues with an edge incident with a, say {a, b}. The edge {a, b} contributes one to deg(a). Each time the circuit passes through a vertex it contributes two to the vertex's degree, because the circuit enters via an edge incident with this vertex and leaves via another such edge. Finally, the circuit terminates where it started, contributing one to deg(a). Therefore, deg(a) must be even, because the circuit contributes one when it begins, one when it ends, and two every time it passes through a (if it ever does). A vertex other than a has even degree because the circuit contributes two to its degree each time it passes through the vertex. We conclude that if a connected graph has a Euler circuit, then every vertex must have even degree. Is this necessary condition for the existence of a Euler circuit also sufficient? That is, must a Euler circuit exist in a connected multigraph if all vertices have even degree? This question can be settled affirmatively with a construction.

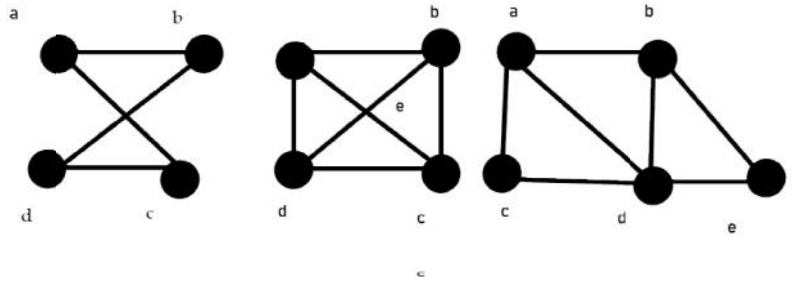


FIGURE 3 The undirected graph $_{G1,G_2}$ and G_3

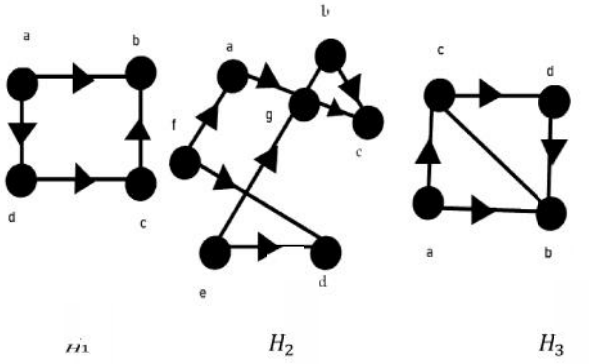


Figure 4: The directed graph $_{H1,}H^2$ and H_3

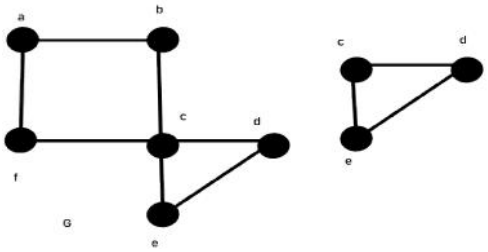


Figure 5 constructing a Euler circuit in G,

Suppose that G is a connected multigraph with at least two vertices and the degree of every vertex of G is even. We will form a simple circuit that begins at an arbitrary vertex a of G, building it edge by edge. Let x_0 = a. First, we arbitrarily choose an edge $\{x_0, x_1\}$ incident with a which is possible because G is connected. We continue by building a simple path $\{x_0, x_1\}$, $\{x_1, x_2\}$ ..., $\{x_{n-1}, x_n\}$, successively adding edges one by one to the path until we cannot add another edge to the path. This happens when we reach a vertex for which we have already included all edges incident with that vertex in the path. For instance, in the graph G in Figure 5 we begin at a and choose in succession the edges {a, f}, {f, c}, {c, b}, and {b, a}. The path we have constructed must terminate because the graph has a finite number of edges, so we are guaranteed to eventually reach a vertex for which no edges are available to add to the path. The path begins at a with an edge of the form {a, x}, and we now show that it must terminate at a with an edge of the form {y, a}. To see that the path must terminate at a, note that each time the path goes through a vertex with even degree, it uses only one edge to enter this vertex, so because the degree must be at least two, at least one edge remains for the path to leave the vertex. Furthermore, every time we enter and leave a vertex of even degree, there are an even number of edges incident with this vertex that we have not yet used in our path. Consequently, as we form the path, every time we enter a vertex other than a, we can leave it. This means that the path can end only at a. Next, note that the path we have constructed may use all the edges of the graph, or it may not if we have returned to a for the last time before

using all the edges. A Euler circuit has been constructed if all the edges have been used. Otherwise, consider the subgraph H obtained from G by deleting the edges already used and vertices that are not incident with any remaining edges. When we delete the circuit a, f, c, b, a from the graph in Figure 5, we obtain the subgraph labeled as H. Because G is connected, H has at least one vertex in common with the circuit that has been deleted. Let w be such a vertex. (In our example, c is the vertex.)

Every vertex in H has even degree (because in G all vertices had even degree, and for each vertex, pairs of edges incident with this vertex have been deleted to form H). Note that H may not be connected. Beginning at w, construct a simple path in H by choosing edges as long as possible, as was done in G. This path must terminate at w. For instance, in Figure 5, c, d, e, c is a path in H. Next, form a circuit in G by splicing the circuit in H with the original circuit in G (this can be done because w is one of the vertices in this circuit). When this is done in the graph in Figure 5, we obtain the circuit a, f, c, d, e, c, b, a. Continue this process until all edges have been used. (The process must terminate because there are only a finite number of edges in the graph.) This produces a Euler circuit. The construction shows that if the vertices of a connected multigraph all have even degree, then the graph has a Euler circuit. We summarize these results in Theorem 1.

Theorem 1 A connected multigraph with at least two vertices has a Euler circuit if and only if each of its vertices has even degree.

We can now solve the Konigsberg bridge problem. Because the multigraph representing these bridges, shown in Figure 2, has four vertices of odd degree, it does not have a Euler circuit. There is no way to start at a given point, cross each bridge exactly once, and return to the starting point. Algorithm 1 gives the constructive procedure for finding Euler circuits given in the discussion preceding Theorem 1. (Because the circuits in the procedure are chosen arbitrarily, there is some ambiguity. We will not bother to remove this ambiguity by specifying the steps of the procedure more precisely.)

Example 3 shows how Euler paths and circuits can be used to solve a type of puzzle.

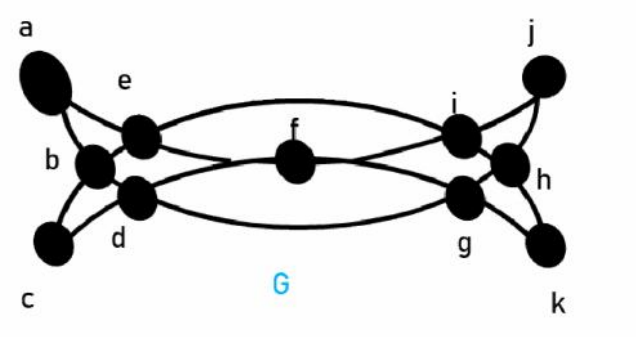


Figure 6 Mohammed 's Scimitars

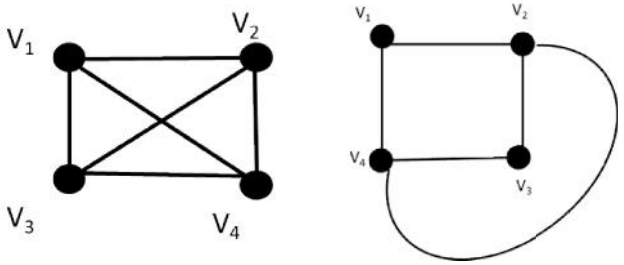
Example 3: Many puzzles ask you to draw a picture in a continuous motion without lifting a pencil so that no part of the picture is retraced. We can solve such puzzles using Euler circuits and paths. For example, can Mohammed's scimitars, shown in Figure 6, be drawn in this way, where the drawing begins and ends at the same point?

Solution: We can solve this problem because the graph G shown in Figure 6 has a Euler circuit. Ithassuchacircuitbecauseallitsverticeshaveevendegree.WewilluseAlgorithm1toconstruct an Euler circuit. First, we form the circuit a, b, d, c, b, e, i, f, e, a. We obtain the subgraph H by deleting the edges in this circuit and all vertices that become isolated when these edges are removed. Then we form the circuit d,g,h,j,i,h,k,g,f,d in H.After forming this circuit we have used all edges in G. Splicing this new circuit into the first circuit at the appropriate place produces the Euler circuit a,b,d,g,h,j,i,h,k, g,f,d,c,b,e,i,f,e,a. This circuit gives a way to draw the scimitars without lifting the pencil or retracing part of the picture.

Another algorithm for constructing Euler circuits, called Fleury's algorithm, is described in the premble to Exercise 50. We will now show that a connected multigraph has a Euler path (and not a Euler circuit) if and only if it has exactly two vertices of odd degree. First, suppose that a connected multigraph doeshaveanEulerpathfroma tob, butnotanEulercircuit.Thefirstedgeofthepathcontributes

one to the degree of a. A contribution of two to the degree of a is made every time the path passes through a. The last edge in the path contributes one to the degree of b. Every time the path goes through b there is a contribution of two to its degree. Consequently, both a and b have odd degree. Every other vertex has even degree, because the path contributes two to the degree of a vertex whenever it passes through it. Now consider the converse. Suppose that a graph has exactly two vertices of odd degree, say a and b. Consider the larger graph made up of the original graph with the addition of an edge {a, b}. Every vertex of this larger graph has even degree, so there is a Euler circuit. The removal of the new edge produces a Euler path in the original graph. Theorem 2 summarizes these results.

Example: Show that the graphs below do not have Euler circuits



(a) (b)

Solution: In graph (a), degree of each vertex is 3. Hence this does not have a Euler circuit. In graph (b), we have

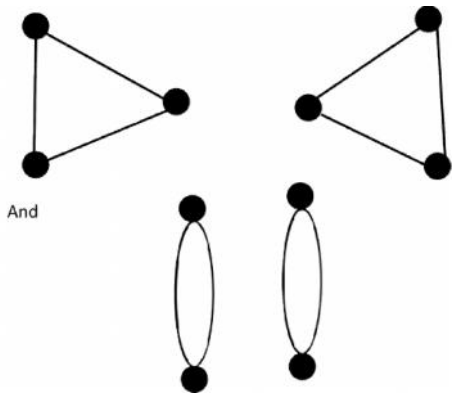
deg(v2) = 3

deg(v4) = 3

Since there are vertices of odd degree in the given graph, therefore it does nothave an Euler circuit.

Remark: The converse of Theorem discussed above is not true. There exist graphs in which every vertex has even degree but the Euler circuits do not exist.

For example,



are graphs in which each vertex has degree 2 but these graphs do not have Euler circuits since there is no path which uses each vertex at least once.

THEOREM 2 A connected multigraph has a Euler path but not a Euler circuit if and only if it has exactly two vertices of odd degree.

OR

If G is a connected graph and every vertex of G has even degree, then G has an Euler circuit.

Proof: Let every vertex of a connected graph G has even degree. If G consists of a single vertex, the trivial walk from v to v is an Euler circuit. So suppose G consists of more than one vertices. We start from any verted v of G. Since the degree of each vertex of G is even, if we reach each vertex other than v by travelling on one edge, the same vertex can be reached by travelling on another previously unused edge. Thus a sequence of distinct adjacent edges can be produced indefinitely as long as v is not reached. Since number of edges of the graph is finite (by definition of graph), the sequence of distinct edges will terminate. Thus the sequence must return to the starting vertex. We

thus obtaina sequence of adjacent vertices and edges starting and ending at v without repeating any edge. Thus we get a circuit C.

If C contains every edge and vertex of G, then C is an Euler circuit.

If C does not contain every edge and vertex of G, remove all edges of C from G and also any vertices that become isolated when the edges of C are removed.

Let the resulting subgraph be G′. We note that when we removed edges of C, an even number of edges from each vertex have been removed. Thus degree of each remaining vertex remains even.

Further since G is connected, there must be at least one vertex common to both C and G′. Let it be w(in fact there are two such vertices). Pick any sequence of adjacent vertices and edges of G′ starting and ending at w without repeating an edge. Let the resulting circuit be C′.

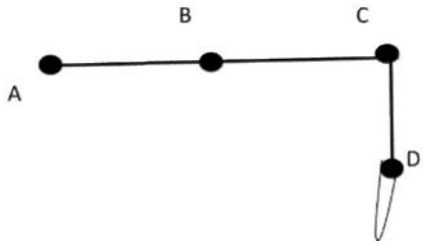
Join C and C′ together to create a new circuit C″. Now, we observe that if we start from v and follow C all the way to reach w and then follow C′ all the way to reach back to w. Then continuing travelling along the untravelled edges of C, we reach v.

If C″ contains every edge an vertex of C, then C″ is an Euler circuit. If not, then we again repeat our process. Since the graph is finite, the process must terminate.

Once again, If G is a connected graph and has exactly two vertices of odd degree, then there is an Euler path in G. Further, any Euler path in G must begin at one vertex of odd degree and end at the other.

Proof: Let u and v be two vertices of odd degree in the given connected graph G. If we add the edge e to G, we get a connected graph G′ all of whose vertices have even degree. Hence there will be an Euler circuit in G′. If we omit e from Euler circuit, we get an Euler path beginning at u(or v) and ending at v(or u).

Examples: Has the graph given below an Eulerian path?



Solution: In the given graph,

deg(A) = 1

deg(B) = 2

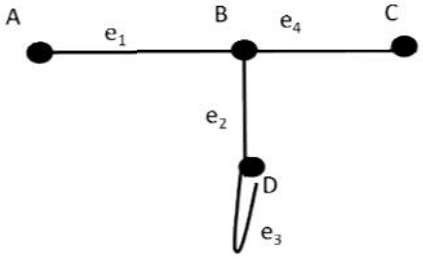
deg(C) = 2

deg(D) = 3

Thus the given connected graph has exactly two vertices of odd degree. Hence, it has an Eulerian path.

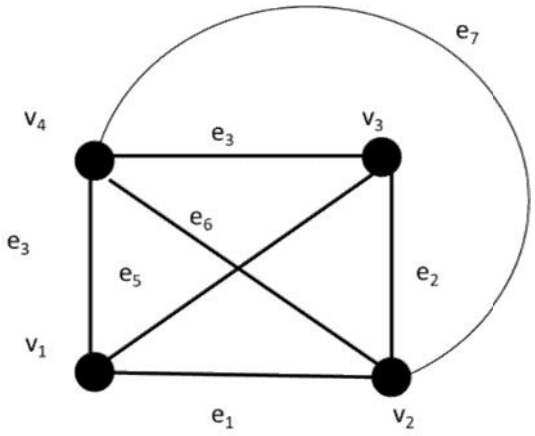
If it starts from A(vertex of odd degree), then it ends at D(vertex of odd degree). If it starts from D(vertex of odd degree), then it ends at A(vertex of odd degree).

But on the other hand if we have the graph as given below :



then deg(A) = 1, deg(B) = 3 deg(C) = 1, degree of D = 3 and so we have four vertices of odd degree. Hence it does not have Euler path.

Example: Does the graph given below possess an Euler circuit?



Solution: The given graph is connected. Further

deg(v1) = 3

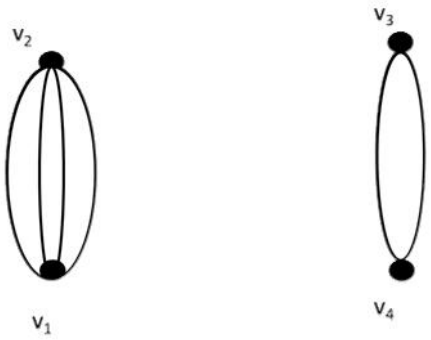
deg(v2) = 4

deg(v3) = 3

deg(v4) = 4

Since this connected graph has vertices with odd degree, it cannot have Eulercircuit. But this graph has Euler path, since it has exactly two vertices of odddegree. For example, v3 e2 v2 e7 v4 e6 v2 e1 v1 e4 v4 e3 v3 e5 v1

Example: Consider the graph



Here, deg(v1) = 4, deg(v2) = 4, deg(v3) = 2, deg(v4) = 2. Thus degree of each vertex is even. But the graph is not Eulerian since it is not connected.

Example 4: Which graphs shown in Figure 7 have a Euler path?

Solution: G_1 containsexactlytwoverticesofodddegree, namely, b and d. Hence, ithasanEuler paththatmusthave b and d asitsendpoints. OnesuchEulerpathis d, a, b, c, d, b. Similarly, G_2 has exactly two vertices of odd degree, namely, b and d. So, it has a Euler path that must have b and d as endpoints. One such Euler path is b, a, g, f, e, d, c, g, b, c, f, d. G3 has no Euler path because it has six vertices of odd degree. ▲

Returning to eighteenth-century Konigsberg, is it possible to start at some point in the town, travel across all the bridges, and end up at some other point in town? This question can

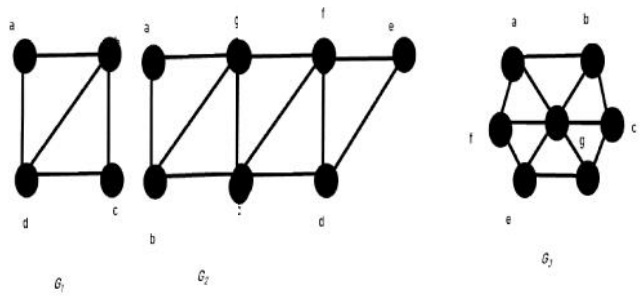


Figure7: Three undirected graphs.

be answered by determining whether there is a Euler path in the multigraph representing the bridges in Konigsberg. Because there are four vertices of odd degree in this multigraph, there is no Euler path, so such a trip is impossible.

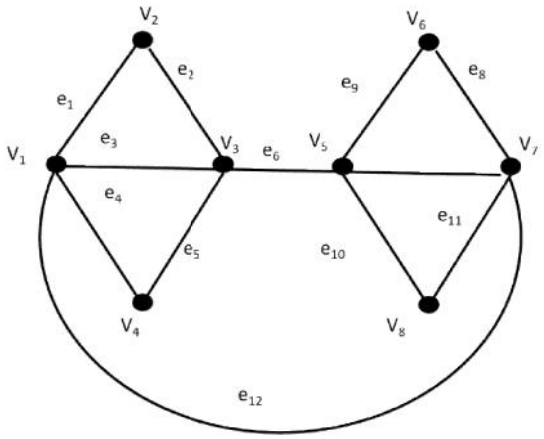
9.3 Applicationsof Euler Pathsand Circuits

Euler paths and circuits can be used to solve many practical problems. For example, many applications ask for a path or circuit that traverses each street in a neighborhood, each road in a transportation network, each connection in a utility grid, or each link in a communications network exactly once. Finding a Euler path or circuit in the appropriate graph model can solve such problems. For example, if a postman can find an Euler path in the graph that represents the streets the postman needs to cover, this path produces a route that traverses each street of the route exactly once. If no Euler path exists, some streets will have to be traversed more than once. The problem of finding a circuit in a graph with the fewest edges that traverses every edge at least once is known as the Chinese postman problem in honor of Guan Meigu, who posed it in 1962. Among the other areas where Euler circuits and paths are applied is in the layout of circuits, in network multicasting, and in molecular biology, where Euler paths are used in the sequencing of DNA.

9.4 Method for Finding Euler Circuit

We know that if every vertex of a non empty connected graph has even degree, then the graph has an Euler circuit. We shall make use of this result to find an Euler path in a given graph.

Consider the graph



We note that

deg(v2) = deg(v4) = deg(v6) = deg(v8) = 2

deg(v1) = deg(v3) = deg(v5) = deg(v7) = 4

Hence all vertices have even degree. Also the given graph is connected. Hence the given has an Euler circuit. We start from the vertex v1 and let C be

C : v1 v2 v3 v1

Then C is not an Euler circuit for the given graph but C intersect the rest of the

graph at v1 and v3. Let C′ be

C′ : v1 v4 v3 v5 v7 v6 v5 v8 v7 v1

(In case we start from v3, then C′ will be v3 v4 v1 v7 v6 v5 v7 v8 v5)

Path C′ into C and obtain

C″ : v 1v2 v3 v1 v4 v3 v5 v7 v6 v5 v8 v7 v1

Or we can write

C″ : e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 e11 e12

(If we had started from v2, then C″ : v1v2 v3 v4 v1 v7 v6 v5 v7 v8 v5 v3 v1 or

e1 e2 e5 e4 e12 e8 e9 e7 e11 e10 e6 e3)

In C″ all edges are covered exactly once. Also every vertex has been covered at least once. Hence C″ is a Euler circuit.

9.5 Hamilton Paths and Circuits

We have developed necessary and sufficient conditions for the existence of paths and circuits that contain every edge of a multigraph exactly once. Can we do the same for simple paths and circuits that contain every vertex of the graph exactly once?

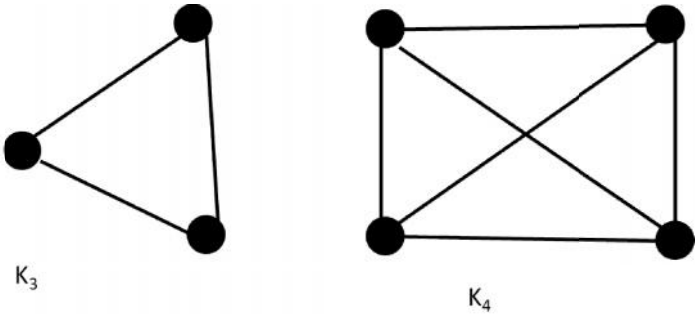
DEFINITION 2 A simple path in a graph G that passes through every vertex exactly once is called a Hamilton path, and a simple circuit in a graph G that passes through every vertex exactly once is called a Hamilton circuit. That is, the simple path x_0, x_1, x_{n-1}, x_n in the graph G = (V, E) is a Hamilton path if V = $\{x_0, x_1, x_{n-1}, x_n\}$and $x_i = x_j$ for $0 \le i < j \le n$, and the simplecircuit $x_0, x_1, x_{n-1}, x_n, x_0$ (with n>0) is a Hamilton circuit if x_0, x_1, x_{n-1}, x_n is a Hamilton path.

Definition: A Hamiltonian Path for a graph G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once.

Definition: A Hamiltonian Circuit for a graph G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and the last which are the same.

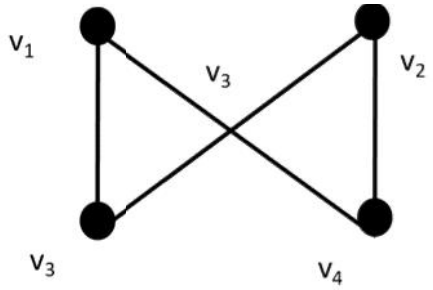
Definition: A graph is called Hamiltonian if it admits a Hamiltonian circuit.

Example 1 : A complete graph Kn has a Hamiltonian Circuit. In particular the Graphs



are Hamiltonian.

Example 2: The graph shown below does not have a Hamiltonian circuit



This terminology comes from a game, called the Icosian puzzle, invented in 1857 by the Irish mathematician Sir William Rowan Hamilton. It consisted of a wooden dodecahedron [a polyhedron with 12 regular pentagons as faces, as shown in Figure 8(a)], with a peg at each vertex of the dodecahedron, and string. The 20 vertices of the dodecahedron were labeled with different cities in the world. The object of the puzzlewas to start at a city and travel along the edges of the dodecahedron, visiting each of the other 19 cities exactly once, and end back at the first city. The circuit traveled was marked off using the string and pegs.

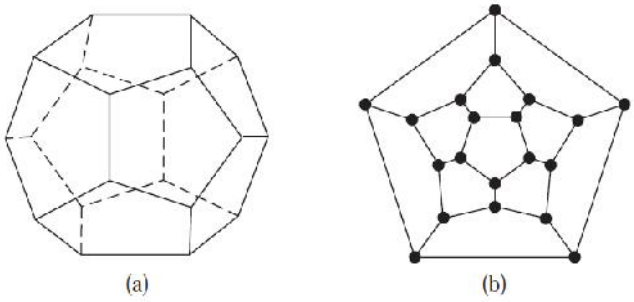


Figure 8 :Hamilton's "A Voyage Round theWorld" Puzzle.

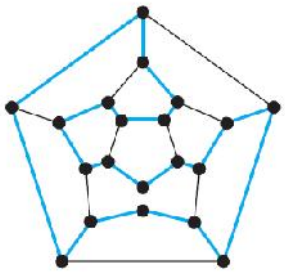


FIGURE 9 A Solution to the "A Voyage Round the World" Puzzle.

Because the author cannot supply each reader with a wooden solid with pegs and string, we will consider the equivalent question: Is there a circuit in the graph shown in Figure 8(b) that passesthrougheachvertexexactlyonce? Thissolvesthepuzzlebecausethisgraphisisomorphic to the graph consisting of the vertices and edges of the dodecahedron. A solution of Hamilton's puzzle is shown in Figure 9.

Example 5: Which of the simple graphs in Figure 10 have a Hamilton circuit or, if not, a Hamilton path?

Solution: G_1 has a Hamilton circuit: a, b, c, d, e, a. There is no Hamilton circuit in G_2 (this can be seen by noting that any circuit containing every vertex must contain the edge{a,b}twice), but G2 does have a Hamilton path, namely, a,b,c,d. G_3 has neither a Hamilton circuit nor a Hamilton path, because any path containing all vertices must contain one of the edges {a,b}, {e,f}, and{c,d}more than once.

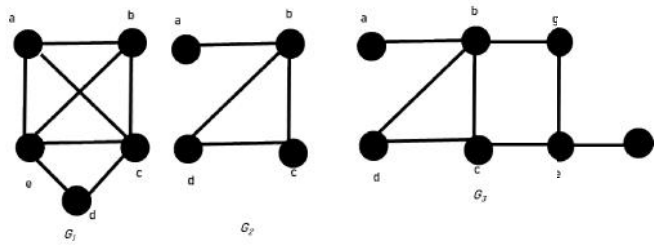


Figure 10: Three simple graphs.

9.6 Conditionsforthe Existenceof Hamilton Circuits

Is there a simple way to determine whether a graph has a Hamilton circuit or path? At first, it might seem that there should be an easy way to determine this, because there is a simple way to answer the similar question of whether a graph has a Euler circuit. Surprisingly, there are no known simple necessary and sufficient criteria for the existence of Hamilton circuits. However, many theorems are known that give sufficient conditions for the existence of Hamilton circuits. Also, certain properties can be used to show that a graph has no Hamilton circuit. For instance, a graph with a vertex of degree one cannot have a Hamilton circuit, because in a Hamilton circuit, each vertex is incident with two edges in the circuit. Moreover, if a vertex in the graph has degree two, then both edges that are incident with this vertex must be part of any Hamilton circuit. Also, note that when a Hamilton circuit is being constructed and this circuit has passed through a vertex, then all remaining edges incident with this vertex, other than the two used in the circuit, can be removed from consideration. Furthermore, a Hamilton circuit cannot contain a smaller circuit within it.

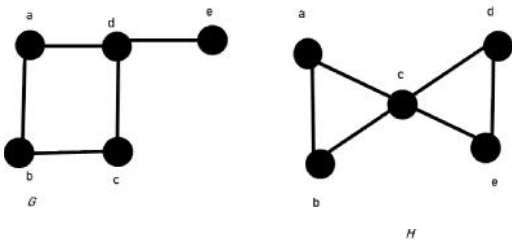


Figure 11: Two Graphs That Do Not Have a Hamilton Circuit.

Solution: There is no Hamilton circuit in G because G has a vertex of degree one, namely, e. Now consider H. Because the degrees of the vertices a, b, d, and e are all two, every edge incident with these vertices must be part of any Hamilton circuit. It is now easy to see that no Hamilton circuit can exist in H, for any Hamilton circuit would have to contain four edges incident with c, which is impossible. ▲

Example7: Show that Kn has a Hamilton circuit whenever n ≥ 3. Solution: We can form a Hamilton circuit in Kn beginning at any vertex. Such a circuit can be built by visiting vertices in any order we choose, as long as the path begins and ends at the same vertex and visits each other vertex exactly once. This is possible because there are edges in Kn between any two vertices. ▲

AlthoughnousefulnecessaryandsufficientconditionsfortheexistenceofHamiltoncircuits are known, quite a few sufficient conditions have been found. Note that the more edges a graph has, the more likely it is to have a Hamilton circuit. Furthermore, adding edges (but not vertices) to a graph with a Hamilton circuit produces a graph with the same Hamilton circuit. So, as we add edges to a graph, especially when we make sure to add edges to each vertex, we make it

Increasingly likely that a Hamilton circuit exists in this graph. Consequently, we would expect there to be sufficient conditions for the existence of Hamilton circuits that depend on the degrees ofverticesbeingsufficientlylarge. Westatetwoofthemostimportantsufficientconditionshere. These conditions were found by Gabriel A. Dirac in 1952 and systein Ore in 1960.

Theorem 3: Dirac'stheorem If G is a simple graph with n vertices with n ≥ 3 such that the degree of every vertex in G is at least n/2, then G has a Hamilton circuit.

Theorem 4:Ore'stheorem If G is a simple graph with n vertices with n ≥ 3 such that deg(u)+deg(v) ≥ n for every pair of nonadjacent vertices u and v in G, then G has a Hamilton circuit.

The proof of Ore's theorem is outlined in Exercise 65. Dirac's theorem can be proved as a corollary to Ore's theorem because the conditions of Dirac's theorem imply those of Ore's theorem. BothOre'stheoremandDirac'stheoremprovidesufficientconditionsforaconnectedsimple graph to have a Hamilton circuit. However, these theorems do not provide necessary conditions for the existence of a Hamilton circuit. For example, the graph C_5 has a Hamilton circuit but does not satisfy the hypotheses of either Ore's theorem or Dirac's theorem, as the reader can verify.

The best algorithms known for finding a Hamilton circuit in a graph or determining that no such circuit exists have exponential worst-case time complexity (in the number of vertices of the graph). Finding an algorithm that solves this problem with polynomial worst-case timecomplexity would be a major accomplishment because it has been shown that this problem is NP-complete. Consequently, the existence of such an algorithm would imply that many other seemingly intractable problems could be solved using algorithms with polynomial worst-case time complexity.

Theorem: Let n be the number of vertices and m be the number of edges in a

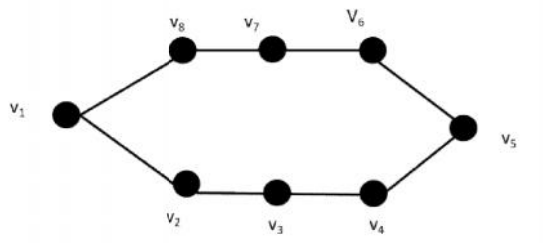
connected graph G. If

m ≥1/2(n2-3n+6).

then G has a Hamiltonian circuit.

The following example shows that the above conditions are not necessary for the existence of Hamiltonian path.

Example : Let G be the connected graph shown in the figure below:



We note that

No. of Vertices in G (n) = 8

No. of Edges in G (m) = 8

Degree of each vertex = 2

Thus, if u and v are non-adjacent vertices, then

deg u + deg v = 2 + 2 = 4 ≥/ 8

Also

$1/2(n^2- 3n +6)=1/2(64-24+6)$

Clearly

$m \geq 1/2(n^2– 3n + 6)$

Therefore the above two theorems fail. But the given graph has Hamiltoniancircuit. For example,

v1 v2 v3 v4, v5, v6, v7, v8, v1

is an Hamiltonian circuit for the graph.

Proposition: Let G be a graph with at least two vertices. If G has a Hamiltonian circuit, then G has a subgraph H with the following properties:

(1) H contains every vertex of G

(2) H is connected

(3) H has the same number of edges as vertices

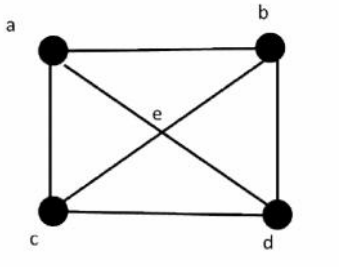
(4) Every vertex of H has degree 2.

The contra positive of this proposition is "If a graph G with at least two vertices does not have a subgraph H satisfying

(1) – (4), then G does not have a Hamiltonian circuit".

Also we know that contrapositive of a statement is logically equivalent to the statement. Therefore the above result can be used to show non-existence of a Hamiltonian Circuit.

Example 1: Does the graph G given below have Hamiltonian circuit?



Solution: The given graph has

No. of vertices (n) = 5

No. of edges (m) = 8

deg(a) = deg(b) = deg(c) = deg(d) = 3

deg(e) = 4

We observe that

(i) degree of each vertex is greater than n/2

(ii) The sum of any non-adjacent pair of vertices is greater than n

(iii) $1/2(n^2- 3n +6)=1/2(25-15+6)=8$

Thus the condition

$m \geq 1/2(n^2-3n+6)$

is satisfied.

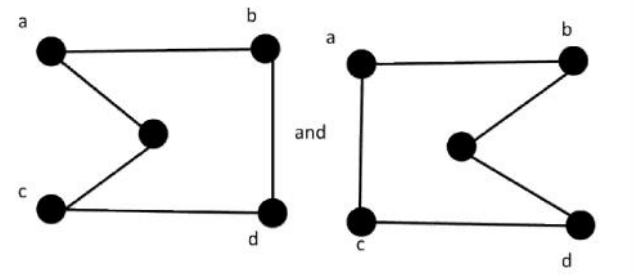
(iv) The sum of degrees of each pair of vertices in the given graph is greater

than $n-1 = 5-1 = 4$.

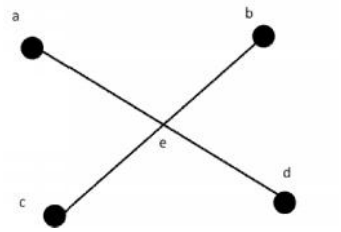
Thus four sufficiency condition are satisfied (whereas one condition out of these four conditions is sufficient for the existence of Hamiltonian path/graph).

Hence the graph has a Hamiltonian Circuit.

For example, the following circuits in G are Hamiltonian:



Example 2 : Does the graph shown below has Hamiltonian circuit?



Solution: Here

No. of vertices (n) = 5

No. of edge (m) = 4

deg(a) = deg(b) = deg(c) = deg(d) = 1

deg(e) = 4

We note that (i) deg(a) = deg(b) = deg(c) = deg(d) $\geq/ 5/2$

(ii) deg(a) + deg(b) = 2 $\geq/$ 5, that is sum of any non-adjacent pair of vertices is not greater than 5

(iii)$1/2(n^2- 3n +6)=1/2(25-15+6)$. Therefore the condition

$m \geq 1/2 (n^2– 3n + 6)$

is not satisfied.

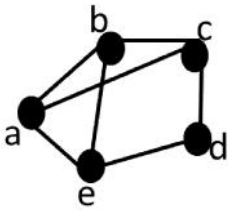
(iv) deg(a) + deg(b) = 2 $\geq/$ 4, i.e., the condition that sum of degrees of each pair of vertices in the graph is not greater than or equal to n–1.

Hence no sufficiency condition is satisfied. So we try the proposition stated above. Suppose that G has a Hamiltonian circuit, then G should a subgraph which contains every vertex of G, and number of vertices and no. of edges in H should be same. Thus H should have 5 vertices a, b, c, d, e and 5 edges. Since G has only 4 edges, H cannot have more than 4 edges. Hence no such subgraph is possible. Hence, the given graph does not have Hamiltonian circuit.

Summary

- We have learned what are Euler path in an undirected graph
- We have learned what are Euler path in a directed and undirected graph.
- We have learned what are Hamilton path and graphs in a directed and undirected graph.

Self Assessment



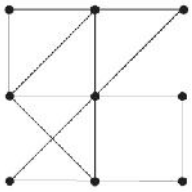
1.Following graph

A. Neither has a Euler circuit nor has a Euler path

B. has a Euler circuit

C. has a Euler path

D. has a Euler circuit and has a Euler Path

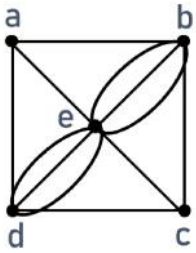
2.Following graph 

A. Neither has a Euler circuit nor has a Euler path

B. has a Euler circuit

C. has only an Euler path

D. has a Euler circuit and but not Euler Path



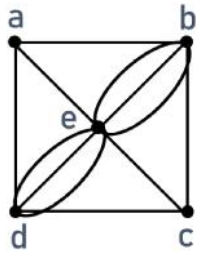
3.Following graph

A. Neither has a Euler circuit nor has a Euler path

B. has a Euler circuit

C. has a Euler path

D. has a Euler circuit and has a Euler Path

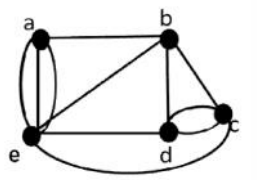
4.Following graph 

A. has a Euler path a, c, c, e, i, e, d, b, u, c, d

B. has a Euler path a, e, c, e, b, e, d, b, a, c, d

C. has a Euler path a, r, c, e, b, y, d, b, c, c, d

D. has a Euler path a, o, c, e, b, e, u, b, a, c, d



5.Following graph euler circuit is

A. a, b, e, d, c, f, d, b, e, a, h, a

B. a, d, c, d, c, e, t, b, e, a, e, a

C. a, b, c, d, c, e, d, b, e, a, e, a

D. a, i, c, d, c, e, o, b, e, u, a, e, a

6.For which values of n do the graphs k_n have a Euler path but no Euler circuit?

A. n is odd

B. n is even

C. n is prime number

D. n is a fractional number

7.For which values of n do the graphs C_n have a Euler path but no Euler circuit?

A. all

B. 5

C. None

D. 7

8.For which values of n do the graph W_n have a Euler path but no Euler circuit?

A. All

B. None

C. 4

D. 3

9.For which values of n do the graphs Q_n have a Euler path but no Euler circuit?

A. N=1

B. N=even

C. N=odd

D. N=prime

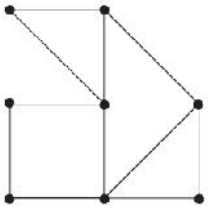
10. When can the center lines of the streets in a city be painted without traveling a street more than once? (Assume that all the streets are two-way streets.)

A. When the graph in which vertices represent intersections and edges streets has an Euler path

B. When the graph in which vertices does not represent intersections and edges streets has an Euler path

C. When the graph in which vertices represent intersections and edges streets does not have an Euler path

D. When the graph in which vertices represent intersections and edges streets have an Euler circuit.

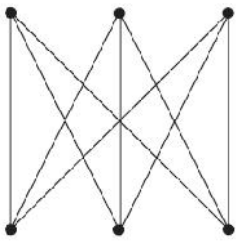
11. graph 

A. neither has a Hamilton path nor a circuit.

B. has a Hamilton circuit.

C. does not have a Hamilton path.

D. does not have a Hamilton circuit.

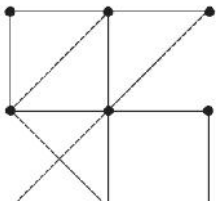
12. The graph 

A. neither has a Hamilton path nor a circuit.

B. does not have a Hamilton path.

C. has a Hamilton circuit.

D. does not have a Hamilton circuit.

13. The graph 

A. has a Hamilton path and a circuit.

B. does not have a Hamilton path.

C. has a Hamilton circuit.

D. does not have a Hamilton circuit.

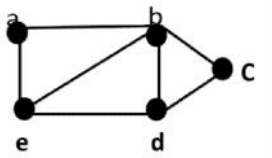
14. For which values of m and n does the complete bipartite graph $K_{m,n}$ have a Hamilton circuit?

A. $m = n \geq 3$

B. $m = n \geq 12$

C. $m = n \geq 2$

D. $m = n \geq 32$

15. 

A. has a Hamilton circuit a, c, d, b, e, a.

B. has a Hamilton circuit a, f, c, i, e, o.

C. has a Hamilton circuit a, f, c, u, e, o.

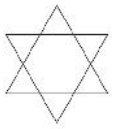
D. has a Hamilton circuit a, b, c, d, e, a.

Answers for Self Assessment

1.	A	2.	B	3.	C	4.	B	5.	C
6.	A	7.	B	8.	B	9.	B	10.	A
11.	B	12.	C	13.	D	14.	C	15.	D

Review Questions

Q1. Suppose that in addition to the seven bridges of Konigsberg(shown in Figure 1) there were two additionalbridges, connecting regions B and C and regions B andD, respectively. Could someone cross all nine of thesebridges exactly once and return to the starting point?



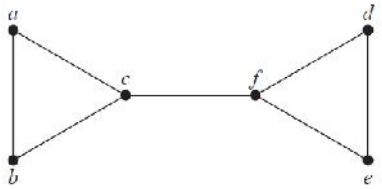
Q2. Determine whether the picture canbe drawn with a pencil in a continuous motion without liftingthe pencil or retracing part of the picture.

Q3. For which values of m and n does the complete bipartite

graph Km,n have an

a) Euler circuit?

b) Euler path?



Q4. Determine whether the given graph has aHamilton circuit. If it does, find such a circuit. If it does not,
give an argument to show why no such circuit exists.

Q5. Does the graph in Q4 have a Hamilton path? Ifso, find such a path. If it does not, give an argument toshow why no such path exists.

Further Readings

1. Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
2. Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.
3. Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.
4. Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 10: Shortest-Path Problems

CONTENTS

Objectives
Introduction
10.1 Dijkstra's Shortest Path Algorithm
10.2 Planar Graphs
10.3 Applications of Planar Graphs
Summary
Self Assessment
Answers for Self Assessment
Review Questions
Further Readings

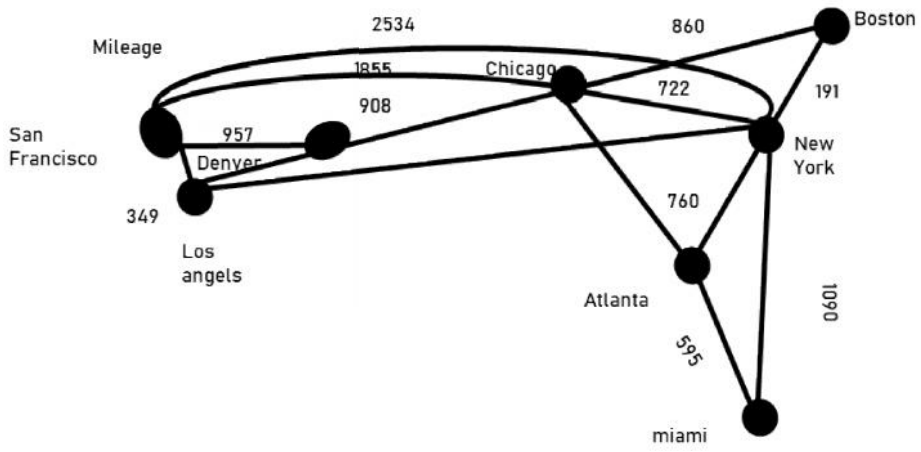
Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- Understand what are Shortest-Path Problems
- Understand what is a Shortest-Path Algorithm.
- Understand what is Dijkstra's Algorithm.
- Understand what are Planar Graphs
- Understand how $K_{3,3}$ is a non-planar Graph

Introduction

Many problems can be modeled using graphs with weights assigned to their edges. As an illustration, consider how an airline system can be modeled. We set up the basic graph model by representing cities by vertices and flights by edges. Problems involving distances can be modeled by assigning distances between cities to the edges. Problems involving flight time can be modeled by assigning flight times to edges. Problems involving fares can be modeled by



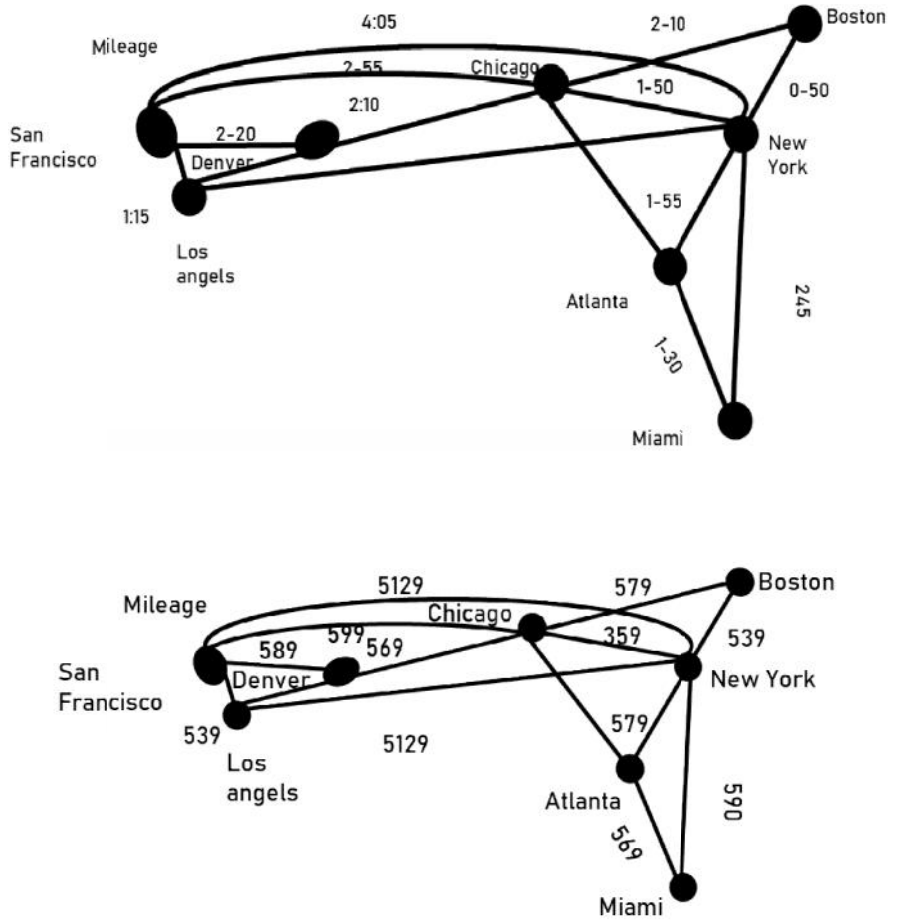
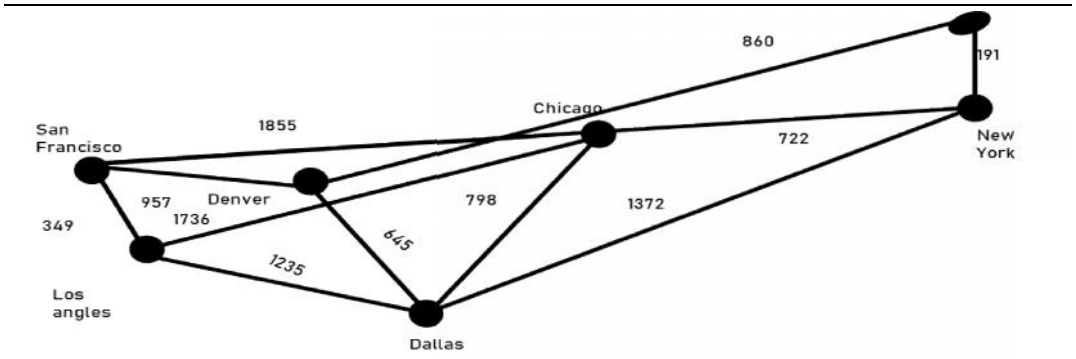
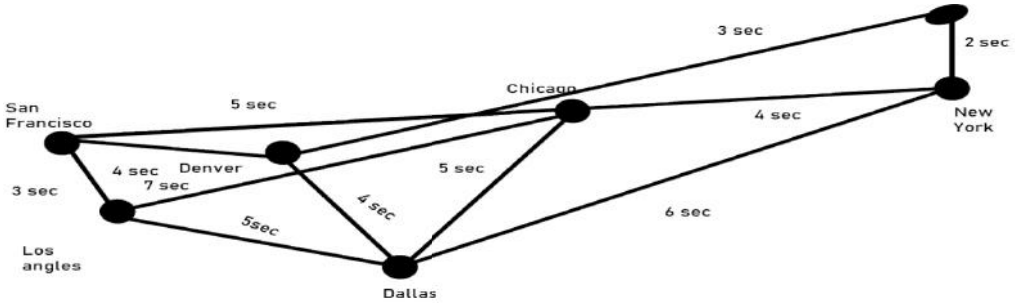


Figure 1: Weighted Graphs Modeling an Airline System.

assigning fares to the edges. Figure 1 displays three different assignments of weights to the edges of a graph representing distances, flight times, and fares, respectively. Graphs that have a number assigned to each edge are called weighted graphs. Weighted graphs are used to model computer networks. Communications costs (such as the monthly cost of leasing a telephone line), the response times of the computers over these lines, or the distance betweencomputers, canallbestudiedusingweightedgraphs. Figure2displays weighted graphs that represent three ways to assign weights to the edges of a graph of a computer network, corresponding to distance, response time, and cost. Several types of problems involving weighted graphs arise frequently. Determining a path of least length between two vertices in a network is one such problem. To be more specific, let the length of a path in a weighted graph be the sum of the weights of the edges of this path. (The reader should note that this use of the term length is different from the use of length to denote the number of edges in a path in a graph without weights.) The question is: What is a shortest path, that is, a path of least length, between two given vertices? For instance, in the airline system



RESPONSE TIME



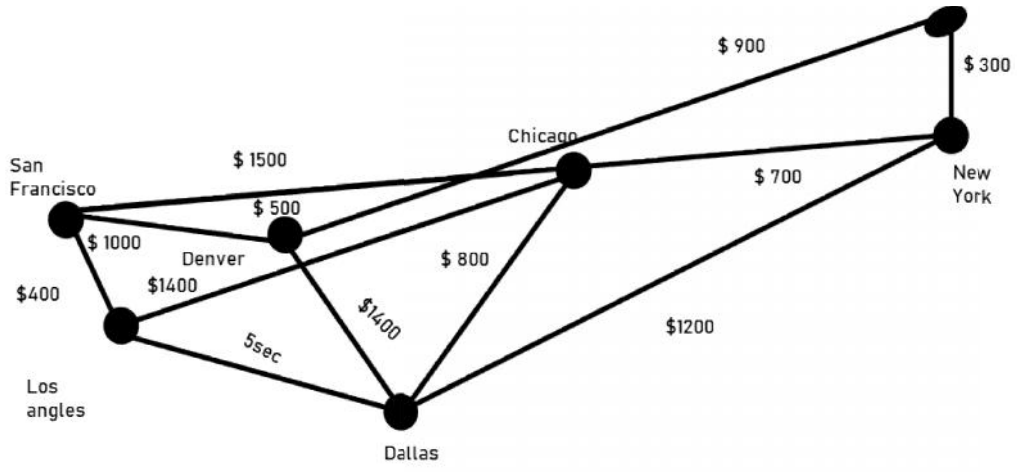


Figure 2: Weighted Graphs Modeling a Computer Network.

Represented by the weighted graph shown in Figure 1, what is a shortest path in air distance between Boston and Los Angeles? What combinations of flights have the smallest total flights time (that is, total time in the air, not including time between flights) between Boston and Los Angeles? What is the cheapest fare between these two cities? In the computer network shown in Figure 2, what is a least expensive set of telephone lines needed to connect the computers in San Francisco with those in New York? Which set of telephone lines gives a fastest response time for communications between San Francisco and New York? Which set of lines has a shortest overall distance? Another important problem involving weighted graphs asks for a circuit of shortest total length that visits every vertex of a complete graph exactly once. This is the famous traveling salesperson problem, which asks for an order in which a salesperson should visit each of the cities on his route exactly once so that he travels the minimum total distance. We will discuss the traveling salesperson problem later in this section.

Let s and t be two vertices of a connected weighted graph G. Shortest Path problem is to find a path from s to t whose total edge weight is minimum.

We now discuss Algorithm due to E. W. Dijkstra which efficiently solve the shortest path problem. The idea is to grow a Dijkstra tree, starting at the vertex s, by adding, at each iteration, a frontier edge, whose non-tree end point is as close as possible to s. The algorithm involves assigning labels to vertices.

For each tree vertex x, let dist [x] denote the distance from vertex s to x and for each edge e in the given weighted graph G, let w(e) be its edge – weight.

After each iteration, the vertices in the Dijkstra tree (the labeled vertices) are those to which the shortest paths from s have been found.

Priority of the Frontier Edges : Let e be a frontier edge and let its P – value be given by

P(e) = dist [x] + w(e),where x is the labeled end point of e and w(e) is the edge – weight of e. Then

(i) The edge with the smallest P – value is given the highest priority.

(ii) The P – value of this highest priority edge e gives the distant from the vertex s to the unlabeled endpoint of e.

We are now in a position to describe Dijkstra shortest path algorithm.

A Shortest-Path Algorithm

There are several differential gorithms that find a shortest path between two vertices in a weighted graph. We will present a greedy algorithm discovered by the Dutch mathematician -

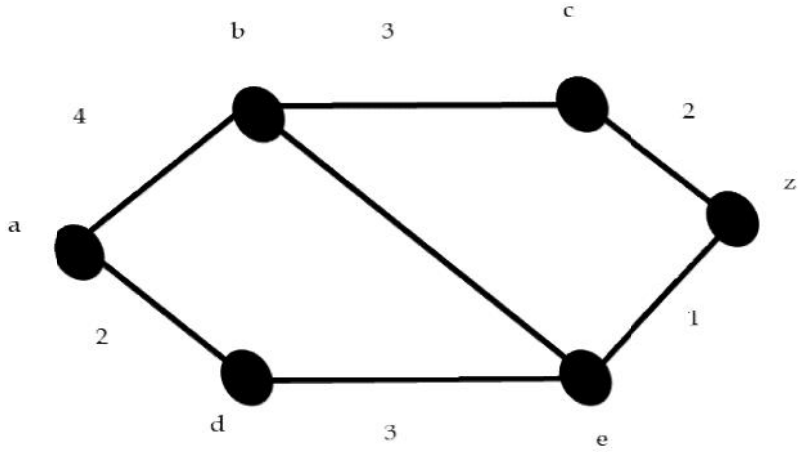


Figure 3 A weighted simple graph

Edsger Dijkstra in 1959. The version we will describe solves this problem in undirected weighted graphs where all the weights are positive. It is easy to adapt it to solve shortest-path problems in directed graphs. Before giving a formal presentation of the algorithm, we will give an illustrative example.

Example 1: What is the length of a shortest path between a and z in the weighted graph shown in Figure 3?

Solution: Although a shortest path is easily found by inspection, we will develop some ideas useful in understanding Dijkstra's algorithm. We will solve this problem by finding the length of a shortest path from a to successive vertices, until z is reached. The only paths starting at a that contain no vertex other than a are formed by adding an edge that has a as one endpoint. These paths have only one edge. They are a, b of length 4 and a, d of length 2. It follows that d is the closest vertex to a, and the shortest path from a to d has length 2. We can find the second closest vertex by examining all paths that begin with the shortest path from a to a vertex in the set {a, d}, followed by an edge that has one endpoint in {a, d}and its other endpoint not in this set. There are two such paths to consider, a, d, e of length 7 and a, b of length 4. Hence, the second closest vertex to a is b and the shortest path from a to b has length 4. To find the third closest vertex to a, we need examine only the paths that begin with the shortest path from a to a vertex in the set {a, d, b},

followed by an edge that has one endpoint in the set {a, d, b} and its other endpoint not in this set. There are three such paths, a, b, c of length 7, a, b, eof length 7, and a, d, eof length 5. Because the shortest of these paths is a, d, e, the third closest vertex to a is e and the length of the shortest path from a to e is 5.

To find the fourth closest vertex to a, we need examine only the paths that begin with the shortest path from a to a vertex in the set{a,d,b,e}, followed by an edge that has one endpoint in the set{a,d,b,e}and its other endpoint not in this set. There are two such paths, a, b, c of length 7 and a, d, e, z of length 6. Because the shorter of these paths is a, d, e, z, the fourth closest vertex to a is z and the length of the shortest path from a to z is 6. ▲

Example1illustratesthegeneralprinciplesusedinDijkstra'salgorithm.Notethatashortest path from a to z could have been found by a brute force approach by examining the length of every path from a to z. However, this brute force approach is impractical for humans and even for computers for graphs with a large number of edges. We will now consider the general problem of finding the length of a shortest path between a and z in an undirected connected simple weighted graph. Dijkstra's algorithm proceeds by finding the length of a shortest path from a to a first vertex, the length of a shortest path from a to a second vertex, and so on, until the length of a shortest path from a to z is found. As a side benefit, this algorithm is easily extended to find the length of the shortest path from a to all other vertices of the graph, and not just to z. The algorithm relies on a series of iterations. A distinguished set of vertices is constructed by adding one vertex at each iteration. A labeling procedure is carried out at each iteration. In this labeling procedure, a vertex w is labeled with the length of a shortest path from a to w that contains only vertices already in the distinguished set. The vertex added to the distinguished set is one with a minimal label among those vertices not already in the set. We now give the details of Dijkstra's algorithm. It begins by labeling a with 0 and the other vertices with ∞. We use the notation $L_0(a) = 0$ and $L_0(v) = \infty$for these labels before any iterations have taken place (the subscript 0 stands for the "0th" iteration). These labels are the lengths of shortest paths from a to the vertices, where the paths contain only the vertex a. (Because no path from a to a vertex different from a exists, ∞is the length of a shortest path between a and this vertex.) Dijkstra's algorithm proceeds by forming a distinguished set of vertices. Let Sydenote this set after k iterations of the labeling procedure. We begin with $S_0 = \emptyset$. The set s_k is formed from $s_{k-1}$by adding a vertex u not in $s_{k-1}$with the smallest label. Once u is added to s_k, we update the labels of all vertices not in s_k, so that Lk(v), the label of the vertex v at the kth stage, is the length of a shortest path from a to v that contains vertices only in s_k (that is, vertices that were already in the distinguished set together with u). Note that the way we choose the vertex u to add to S_k at each step is an optimal choice at each step, making this a greedy algorithm. (We will prove shortly that this greedy algorithm always produces an optimal solution.) Let v be a vertex not in S_k.To update the label of v, note that Lk(v) is the length of a shortest path from a to v containing only vertices in s_k The updating can be carried out efficiently when this observation is used: A shortest path from a to v containing only elements of Sk is either a shortest path from a to v that contains only elements of S_{k-1} (that is, the distinguished vertices not including u), or it is a shortest path from a to u at the (k−1)st stage with the edge {u,v} added. In other words,

$L_k (a, v) = \min \{L_k-1(a, v), L_k -1(a, u) + w (u, v)\}$,

where w (u, v) is the length of the edge with u and v as endpoints. This procedure is iterated by successively adding vertices to the distinguished set until z is added. When z is added to the distinguished set, its label is the length of a shortest path from a to z. Dijkstra's algorithm is given in Algorithm 1. Later we will give a proof that this algorithm is correct. Note that we can find the length of the shortest path from a to all other vertices of the graph if we continue this procedure until all vertices are added to the distinguished set.

10.1 Dijkstra's Shortest Path Algorithm

Input : A connected weighted graph G with non-negative edge-weights and a vertex s of G.

Output : A spanning tree T of G, rooted at the vertex s, whose path from s to each vertex v is a shortest path from s to v in G and a vertex labeling giving the distance from s to each vertex.

Initialize the Dijkstra tree T as vertex s.

Initialize the set of frontier edges for the tree T as empty.

dist : [s] = 0.

Write label 0 on vertex s.

While Dijkstra tree T does not yet span G

For each frontier edge e for T,

Let x be the labeled endpoint of edge e.

Let y be the unlabeled endpoint of edge e.

Set P(e) = dist [x] + w(e)

Let e be a frontier edge for T that has smallest P – value

Let x be the labeled endpoint of edge e

Let y be the unlabeled endpoint of edge e

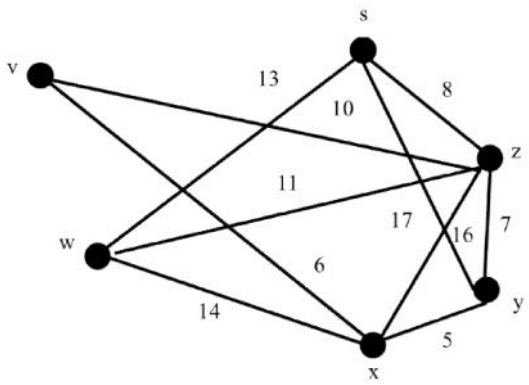
Add edge e (and vertex y) to tree T

dist [y] : P(e)

Write label dist [y] on vertex y.

Return Dijkstra tree T and its vertex labels.

Example: Apply Dijkstra algorithm to find shortest path from s to each of the vertex in the graph given below

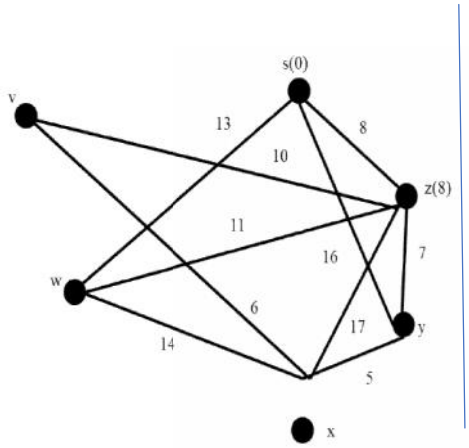


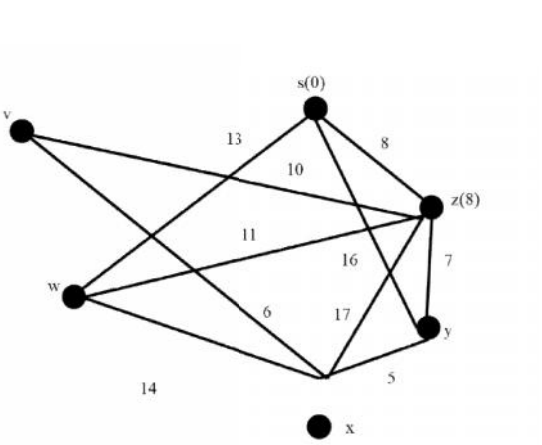
If t is the labeled endpoint of edge e, then P – values are given by

P(e) = dist [t] + w(e),

where dist [t] = distance from s to t and w(e) is the edge weight of edge e. For each vertex v, dist [v] appears in the parenthesis. Iteration tree at the end of each iteration is drawn in dark line

Iteration 1 Iteration 2





dist [s] = 0 P(sw) = 13 (minimum) dist [s] = 0 P(zy) = 8 + 7 =15

P(sy) = 16 dist [z] = 8 P(zx) = 8 + 17 = 25

P(zv) = 8 + 10 = 18

P(zw) = 8 + 11 = 19

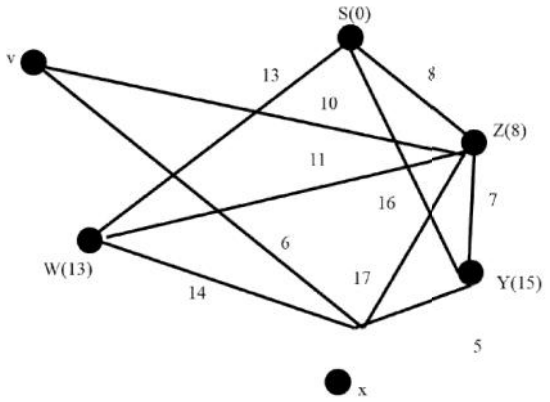
P(zx) = 8 + 17 = 25

dist [w] = 13 P(zv) = 8 + 10 = 18

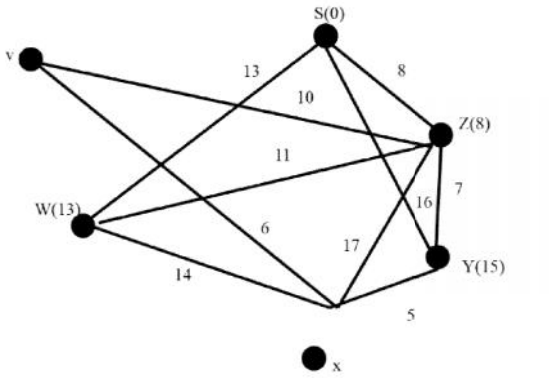
P(sy) = 16

P(wx) = 13 + 14 = 27

Iteration 3



iteration 4



dist [s] = 0 P(zv) = 18 (minimum)

(minimum)

dist [z] = 8 P(zx) = 8 + 17 = 15

dist [w]= 13 P(wx) = 13 + 14 = 27

dist [y] = 15 P(yx) = 15 + 5 = 20

Dist [s] = 0 P(yx) = 20

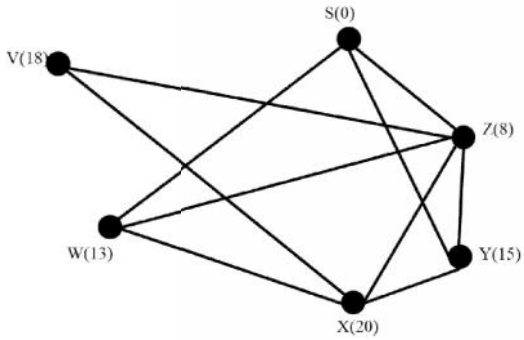
Dist [z] = 8 P(zx) = 8 + 17 = 25

Dist [w] = 13 P(vx) = 18 + 6 =14

Dist [y] = 15 P(wx) = 13 + 14 =27

Dist [v] = 18

Iteration 5



dist [s] = 0 • x (20)

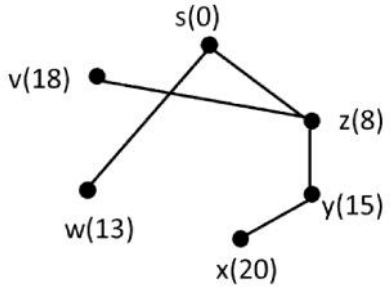
dist [z] = 8

dist [w] =13

dist [y] =15

dist [x] = 20

which are the required shortest paths from s to any other point. The Dijkstra tree is shown in dark lines.



Example 2 illustrates how Dijkstra's algorithm works. Afterward, we will show that this algorithm always produces the length of a shortest path between two vertices in a weighted graph.

Example 2: Use Dijkstra's algorithm to find the length of a shortest path between the vertices a and z in the weighted graph displayed in Figure 4(a).

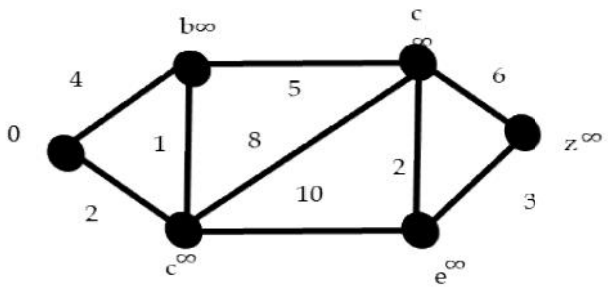
Solution: The steps used by Dijkstra's algorithm to find a shortest path between a and z are shown in Figure 4. At each iteration of the algorithm the vertices of the set Sk are circled. A shortest path from a to each vertex containing only vertices in Sk is indicated for each iteration. The algorithm terminates when z is circled. We find that a shortest path from a to z is a, c, b, d, e, z, with length 13. ▲

Remark: In performing Dijkstra's algorithm it is sometimes more convenient to keep track of labels of vertices in each step using a table instead of redrawing the graph for each step.

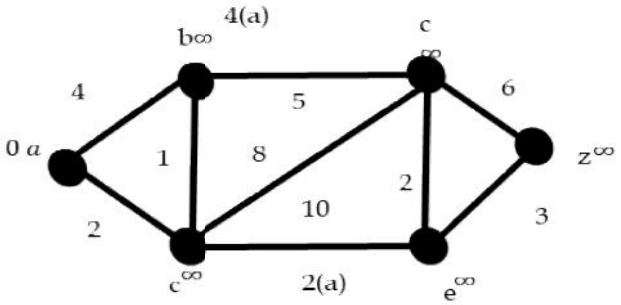
Next, we use an inductive argument to show that Dijkstra's algorithm produces the length of a shortest path between two vertices a and z in an undirected connected weighted graph. Take as the inductive hypothesis the following assertion: At the kth iteration

(i) the label of every vertex v in S is the length of a shortest path from a to this vertex, and (ii) the label of every vertex not in S is the length of a shortest path from a to this vertex that contains only (besides the vertex itself) vertices in S.

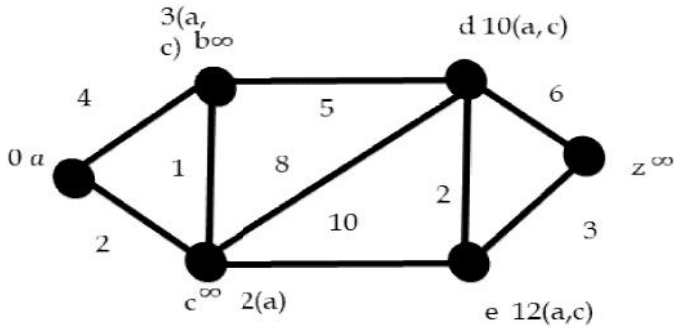
When k = 0, before any iterations are carried out, S =∅, so the length of a shortest path from a to a vertex other than a is∞. Hence, the basis case is true.



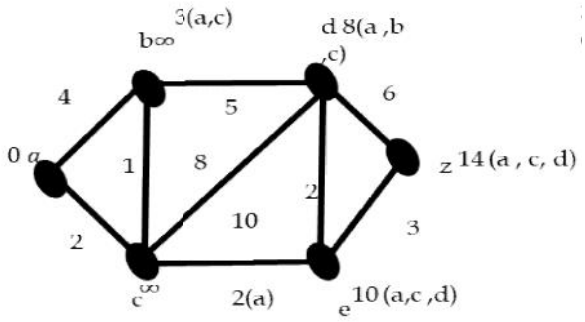
(a)


4(a)
b∞
c
4
5
6
0 a
1
8
z∞
2
10
3
2
c∞
2(a)
e∞

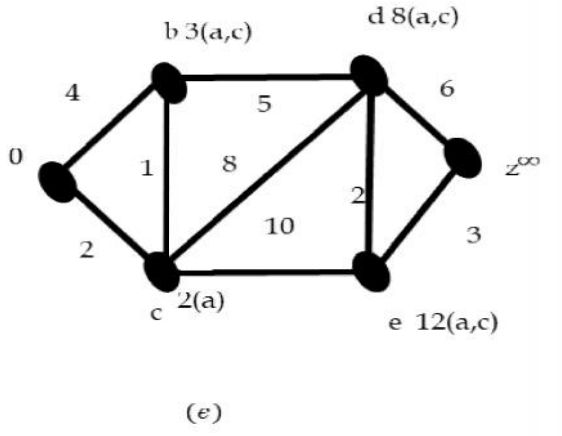
(b)


3(a,
c) b∞
d 10(a, c)
4
5
6
0 a
1
8
z ∞
2
10
3
2
c∞ 2(a)
e 12(a,c)

(c)


3(a,c)
b∞
d 8(a ,b
,c)
4
5
6
0 a
1
8
z 14 (a , c, d)
2
10
3
2
c∞
2(a)
e 10 (a,c ,d)

(d)


b 3(a,c)
d 8(a,c)
4
5
6
0
1
8
z∞
2
10
3
2
c 2(a)
e 12(a,c)

(e)

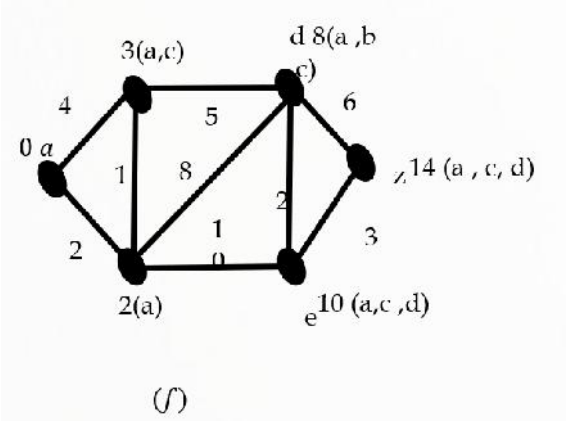


FIGURE 4 Using Dijkstra's Algorithm to Find a Shortest Path from a to z.

Assume that the inductive hypothesis holds for the kth iteration. Let v be the vertex added to S at the (k+1) st iteration, so v is a vertex not in S at the end of the kth iteration with the smallest label (in the case of ties, any vertex with smallest label may be used). From the inductive hypothesis we see that the vertices in S before the (k+1) st iteration are labeled with the length of a shortest path from a. Also, v must be labeled with the length of a shortest path to it from a. If this were not the case, at the end of the kth iteration there would be a path of length less than $L_k(v)$ containing a vertex not in S [because $L_k(v)$ is the length of a shortest path from a to v containing only vertices in S after the kth iteration]. Let u be the first vertex not in S in such a path. There is a path with length less than $L_k(v)$ from a to u containing only vertices of S. This contradicts the choice of v. Hence, (i) holds at the end of the (k+1) st iteration. Let u be a vertex not in S after k+1 iteration. A shortest path from a to u containing only elements of S either contains v or it does not. If it does not contain v, then by the inductive hypothesis its length is $L_k(u)$. If it does contain v, then it must be made up of a path from a to v of shortest possible length containing elements of S other than v, followed by the edge from v to u. In this case, its length would be $L_k(v)+w(v, u)$. This shows that (ii) is true, because $L_k+1(u) = \min\{L_k(u),L_k(v)+w(v,u)\}$. We now state the theorem that we have proved.

Theorem 1: Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

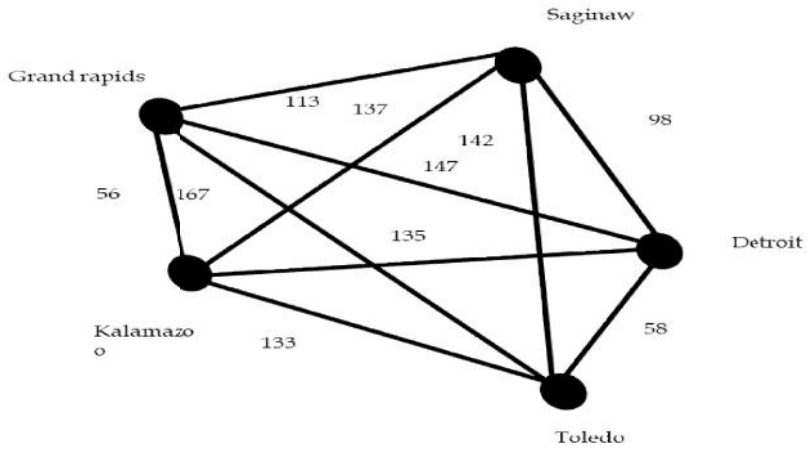
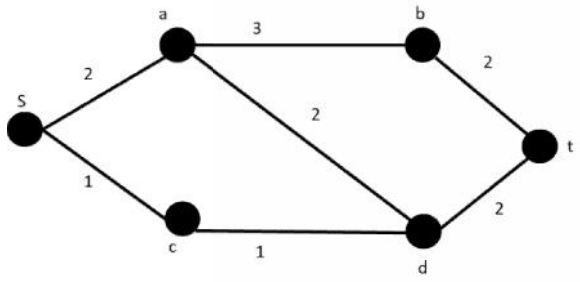


FIGURE 5 The graph showing the distances between five cities

We can now estimate the computational complexity of Dijkstra's algorithm (in terms of additions and comparisons). The algorithm uses no more than $n-1$ iterations where n is the number of vertices in the graph, because one vertex is added to the distinguished set at each iteration. We are done if we can estimate the number of operations used for each iteration. We can identify the vertex not in S_k with the smallest label using no more than $n-1$ comparisons. Then we use an addition and a comparison to update the label of each vertex not in S_k. It follows that no more than $2(n-1)$ operations are used at each iteration, because there are no more than $n-1$ labels to update at each iteration. Because we use no more than $n-1$ iterations, each using no more than $2(n-1)$ operations, we have Theorem 2.

Example: Find a shortest path from s to t and its length for the graph given below:



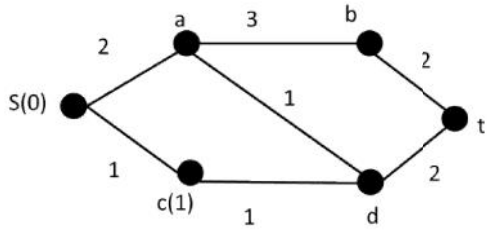
Solution: Let x be the labeled endpoint of edge e, then P-values are given by

P(c) = dist [x] + w(e),

where dist [x] denotes the distance from s to x and w(e) is the weight of the edge e.

For each vertex v, dist [v] appears in the bracket. Iteration tree at the end of each iteration is shown in dark lines.

iteration 1



dist [s] = 0 P(c d) = 2

dist [c] = 1 P(s a) = 2 (minimum)

(minimum)

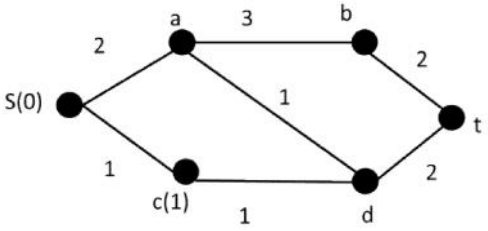
P(s b) = ∞

P(s t) = ∞

P(d t) = ∞

P(b t) = ∞

iteration 2



dist [s] =0 P(a b) = 2 + 3 = 5

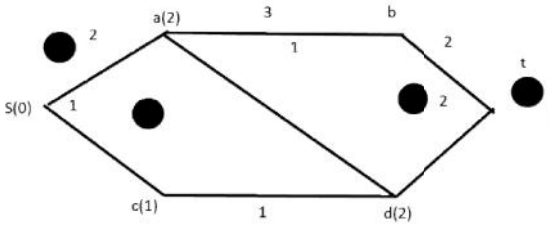
dist [c] = 1 P(c d) = 1 + 1 = 2

dist [a] = 2 P(d t) = ∞

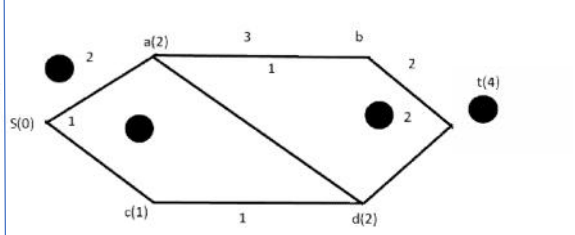
P(b t) = ∞

P(a d) = 3

Iteration 3



iteration 4



dist [s] = 0 P(d t) = 2 (minimum)

dist [c] = 1 P(a b) = 5

dist [a] = 2 P(b t) = ∞

dist [d] = 2

dist [s] =0 P(a b) = 5

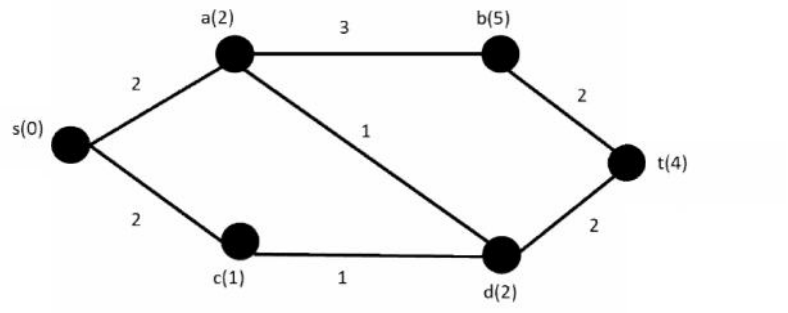
dist [c] = 1 P(b t) = ∞

dist [a] = 2

dist [d] = 2

dist [t] =4

iteration 5



dist [s] =0

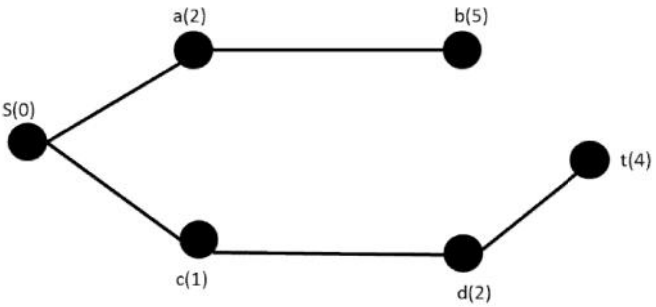
dist [c] =1

dist [a] = 2

dist [d] = 2

dist [t] = 4

dist [b] = 5



Thus, the Dijkstra tree isas above.Thus the shortest path is scat and its length is 4.

Shortest Path if all Edges Have Length 1

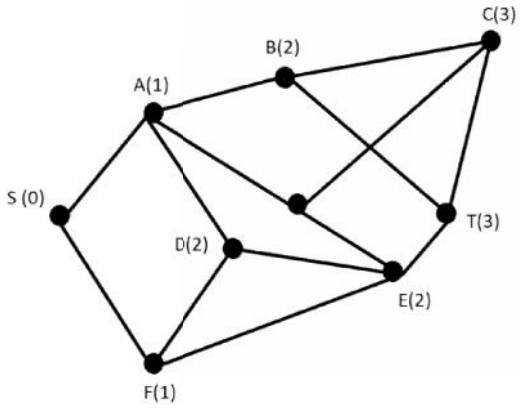
If all edges in a connected graph G have length 1, then a shortest path v1 → vk is the path that has the smallest number of edges among all paths v1 → vk in the given graph G. Moore's Breadth First Search Algorithm This method of finding shortest path in a connected graph G from a vertex s to a vertex t is used when all edges have length 1.

Input : Connected graph G = (V, E), in which one vertex is denoted by s and one by t and each edge (vi , vk) has length 1. Initially all vertices are unlabeled.

Output : A shortest path s → t in G = (V, E). 1. Label s with 0. 2. Set vi = 0 3. Find all unlabeled vertices adjacent to a vertex labeled vi . 4. Label the vertices just found with vi+1 5. If vertex t is labeled, then "back tracking" gives the shortest path. If k is level of t(i.e., t = vk), then Output : vk, vk-1, ……v1, 0. Else increase i by 1. Go to step 3.

End Moore.

Remark : There could be several shortest path from s to t. Example : Use B F S algorithm to find shortest path from s to t in the connected graph G given below:



Solution: Label s with 0 and then label the adjacent vertices with 1. Thus two vertices have been labeled by 1. Now Label the adjacent vertices of all vertices labeled by 1 with label 2. Thus three vertices have been labeled with 2. Label the vertices adjacent to these vertices (labeled by 2) with 3. Thus two vertices have been labeled with 3. We have reached t. Now back tracking yields the

following shortest paths

t(3), e(2), f(1), s(0), that is, s f e t

or

t(3), b(2), a(1), s(0), that is s a b t

or

t(3), e(2), a(1), s(0), that is, s a e t

Thus there are three possible shortest paths of length 3.

Theorem 2: Dijkstra's algorithm uses $O(n^2)$ operations (additions and comparisons) to find the length of a shortest path between two vertices in a connected simple undirected weighted graph with n vertices.

The Traveling Salesperson Problem

We now discuss an important problem involving weighted graphs. Consider the following problem: A traveling salesperson wants to visit each of n cities exactly once and return to his starting point. For example, suppose that the salesperson wants to visit Detroit, Toledo, Saginaw, Grand Rapids, and Kalamazoo (see Figure 5). In which order should he visit these cities to travel the minimum total distance? To solve this problem, we can assume the salesperson starts in Detroit (because this must be part of the circuit) and examine all possible ways for him to visit the other four cities and then return to Detroit (starting elsewhere will produce the same circuits). There is a total of 24 such circuits, but because we travel the same distance when we travel a circuit in reverse order, we need only consider 12 different circuits to find the minimum total distance he must travel. We list these 12 different circuits and the total distance traveled for each circuit. As can be seen from the list, the minimum total distance of 458 miles is traveled using the circuit Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit (or its reverse). n vertices.

We just described an instance of the traveling salesperson problem. The traveling sales person problem asks for the circuit of minimum total weight in a weighted, complete, undirected graph that visits each vertex exactly once and returns to its starting point. This is equivalent to

An 1832 handbook Der Handlungsreisende (The Traveling Salesman) mentions the traveling salesman problem, with sample tours through Germany and Switzerland.

Route	Total Distance (miles)
Detroit–Toledo–GrandRapids–SaginawKalamazoo–Detroit	610
Detroit–Toledo–Grand Rapids–Kalamazoo–Saginaw–Detroit	516
Detroit–Toledo–Kalamazoo–Saginaw–Grand Rapids–Detroit	588
Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit	458
Detroit–Toledo–Saginaw–Kalamazoo–Grand Rapids–Detroit	540
Detroit–Toledo–Saginaw–Grand Rapids–Kalamazoo–Detroit	504
Detroit–Saginaw–Toledo–Grand Rapids–Kalamazoo–Detroit	598
Detroit–Saginaw–Toledo–Kalamazoo–Grand Rapids–Detroit	576
Detroit–Saginaw–Kalamazoo–Toledo–Grand Rapids–Detroit	682
Detroit–Saginaw–Grand Rapids–Toledo–Kalamazoo–Detroit	646
Detroit–Grand Rapids–Saginaw–Toledo–Kalamazoo–Detroit	670
Detroit–Grand Rapids–Toledo–Saginaw–Kalamazoo–Detroit	728

Asking for a Hamilton circuit with minimum total weight in the complete graph, because each vertex is visited exactly once in the circuit. The most straightforward way to solve an instance of the traveling salesperson problem is to examine all possible Hamilton circuits and select one of minimum total length. How many circuits do we have to examine to solve the problem if there are n vertices in the graph? Once a starting point is chosen, there are (n−1)! different Hamilton circuits to examine, because there are n−1 choices for the second vertex, n−2 choices for the third vertex, and so on. Because a Hamiltoncircuitcanbetraveledinreverseorder, weneedonlyexamine (n−1)!/2circuitstofind our answer. Notethat (n−1)!/2growsextremelyrapidly.Tryingtosolveatravelingsalesperson problem in this way when there are only a few dozen vertices is impractical. For example, with 25 vertices, a total of 24! /2 (approximately 3.1×1023) different Hamilton circuits would have to be considered. If it took just one nanosecond (10−9 second) to examine each Hamilton circuit, a total of approximately ten million years would be required to find a minimum-length Hamilton circuit in this graph by exhaustive search techniques. Because the traveling salesperson problem has both practical and theoretical importance, a great deal of effort has been devoted to devising efficient algorithms that solve it. However, no algorithm with polynomial worst-case time complexity is known for solving this problem. Furthermore, if a polynomial worst-case time complexity algorithm were discovered for the traveling salesperson problem, many other difficult problems would also be solvable using polynomial worst-case time complexity algorithms (such as determining whether a proposition in n variables is a tautology, discussed in Chapter 1). This follows from the theory of NPcompleteness. (For more information about this, consult [GaJo79].) A practical approach to the traveling salesperson problem when there are many vertices to visit is to use an approximation algorithm. These are algorithms that do not necessarily produce the exact solution to the problem but instead are guaranteed to produce a solution that is close to an exact solution. (Also, see the preamble to Exercise 46 in the Supplementary Exercises of Chapter 3.) That is, they may produce a Hamilton circuit with total weight We such that W ≤ W≤ cW, where W is the total length of an exact solution and c is a constant. For example, there is an algorithm with polynomial worst-case time complexity that works if the weighted graph satisfies the triangle inequality such that c = 3/2. For general weighted graphs for every positive real number k no algorithm is known that will always produce a solution at most k times a best solution. If such an algorithm existed, this would show that the class P would be the same as the class NP, perhaps the most famous open question about the complexity of algorithms.

In practice, algorithms have been developed that can solve traveling salesperson problems with as many as 1000 vertices within 2% of an exact solution using only a few minutes of computer time.

10.2 Planar Graphs

Introduction

Consider the problem of joining three houses to each of three separate utilities, as shown in Figure 1. Is it possible to join these houses and utilities so that none of the connections cross? This problem can be modeled using the complete bipartite graph K3,3. The original question can be rephrased as: Can K3,3 be drawn in the plane so that no two of its edge's crosses? In this section we will study the question of whether a graph can be drawn in the plane without edges crossing. In particular, we will answer the houses-and-utilities problem. There are always many ways to represent a graph. When is it possible to find at least one way to represent this graph in a plane without any edges crossing?

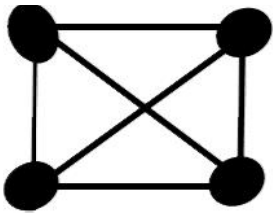


FIGURE 2 The Graph K_4.

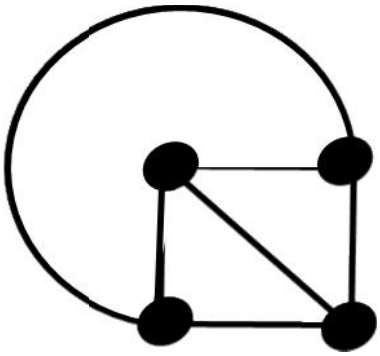


FIGURE 3 K_4 Drawn with No Crossings.

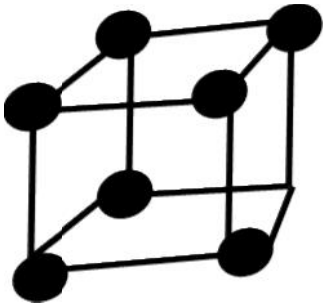


FIGURE 4 The Graph Q_3.

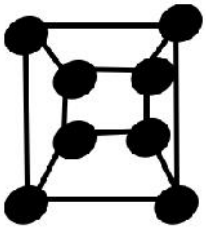


FIGURE 5 A Planar Representation of Q_3.

DEFINITION 1 A graph is called planar if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point other than their common endpoint). Such a drawing is called a planar representation of the graph.

A graph may be planar even if it is usually drawn with crossings, because it may be possible to draw it in a different way without crossings. Example 1: Is K4 (shown in Figure 2 with two edges crossing) planar?

Solution: K4 is planar because it can be drawn without crossings, as shown in Figure 3.

Example 2: Is Q3, shown in Figure 4, planar?

Solution: Q3 is planar, because it can be drawn without any edges crossing, as shown in Figure 5. ▲

We can show that a graph is planar by displaying a planar representation. It is harder to show that a graph is nonplanar. We will give an example to show how this can be done in an ad hoc fashion. Later we will develop some general results that can be used to do this.

Example 3: Is K3,3, shown in Figure 6, planar?

Solution: Any attempt to draw K3,3 in the plane with no edges crossing is doomed. We now show why. In any planar representation of K3,3, the vertices v1 and v2 must be connected to both v4 and v5. These four edges form a closed curve that splits the plane into two regions, R1 and R2, as shown in Figure 7(a). The vertex v3 is in either R1 or R2. When v3 is in R2, the inside of the closed curve, the edges between v3 and v4 and between v3 and v5 separate R2 into two subregions, R21 and R22, as shown in Figure 7(b)

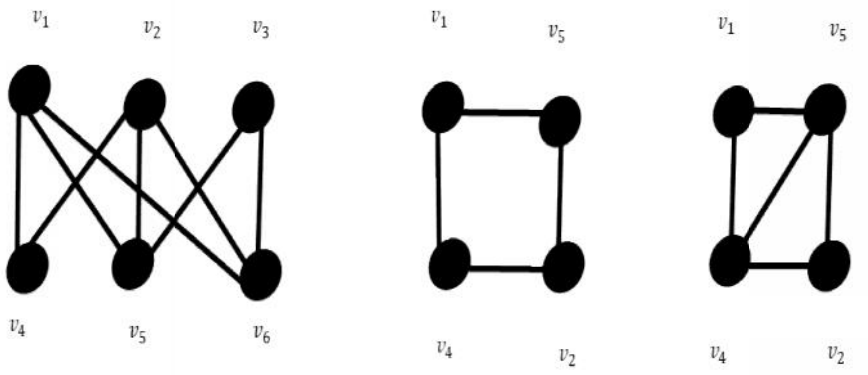


Figure 6: The Graph K3,3. FIGURE 7 Showing that K3,3 Is Nonplanar.

Next, note that there is no way to place the final vertex v6 without forcing a crossing. For if v6 is in R1, then the edge between v6 and v3 cannot be drawn without a crossing. If v6 is in R21, then the edge between v2 and v6 cannot be drawn without a crossing. If v6 is in R22, then the edge between v1 and v6 cannot be drawn without a crossing. A similar argument can be used when v3 is in R1. It follows that K3,3 is not planar. ▲

Example 3 solves the utilities-and-houses problem that was described at the beginning of this section. The three houses and three utilities cannot be connected in the plane without a crossing. A similar argument can be used to show that K5 is nonplanar.

10.3 Applications of Planar Graphs

Planarity of graphs plays an important role in the design of electronic circuits. We can model a circuit with a graph by representing components of the circuit by vertices and connections between them by edges. We can print a circuit on a single board with no connections crossing if the graph representing the circuit is planar. When this graph is not planar, we must turn to more expensive options. For example, we can partition the vertices in the graph representing the circuit into planar subgraphs. We then construct the circuit using multiple layers. (See the preamble to Exercise 30 to learn about the thickness of a graph.) We can construct the circuit using insulated wires whenever connections cross. In this case, drawing the graph with the fewest possible crossings is important. (See the preamble to Exercise 26 to learn about the crossing number of a graph.) The planarity of

graphs is also useful in the design of road networks. Suppose we want to connect a group of cities by roads. We can model a road network connecting these cities using a simple graph with vertices representing the cities and edges representing the highways connecting them. We can build this road network without using underpasses or overpasses if the resulting graph is planar. Euler's Formula

A planar representation of a graph splits the plane into regions, including an unbounded region. For instance, the planar representation of the graph shown in Figure 8 splits the plane into six regions. These are labeled in the figure. Euler showed that all planar representations of a graph split the plane into the same number of regions. He accomplished this by finding a relationship amongthenumberofregions, thenumberofvertices, andthenumberofedgesofaplanargraph.

THEOREM 1 EULER'S FORMULA Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then r = e−v+2.

Proof: First, we specify a planar representation of G. We will prove the theorem by constructing a sequence of subgraphs G_1, G_2, $G_e = G$, successively adding an edge at each stage. This is done using the following inductive definition. Arbitrarily pick one edge of G to obtain G1. ObtainG_n fromG_{n-1} byarbitrarilyaddinganedgethatisincidentwithavertexalreadyinG_{n-1},

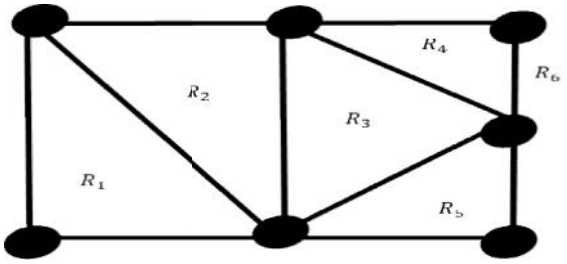


Figure 8: The Regions of the Planar Representation of a Graph.

adding the other vertex incident with this edge if it is not already in Gn−1. This construction is possible because G is connected. G is obtained after e edges are added. Let r_n, e_n, and v_n represent the number of regions, edges, and vertices of the planar representation of Gn induced by the planar representation of G, respectively.

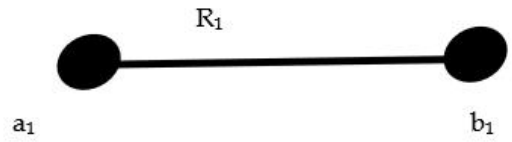


Figure 9: The Basis Case of the Proof of Euler's Formula.

The proof will now proceed by induction. The relationship $r_1 = e_1 - v_1 + 2$ is true for G_1, because $e_1 = 1$, $v_1 = 2$, and $r_1 = 1$. This is shown in Figure 9. Now assume that $r_k = e_k - v_k + 2$. Let $\{a_{k+1}, b_{k+1}\}$ be the edge that is added to $G_k$toobtain G_{k+1}. There are two possibilities to consider. In the first case, both a_{k+1} and $b_{k+1}$are already in G_k. These two vertices must be on the boundary of a common region R, or else it would be impossible to add the edge $\{a_{k+1}, b_{k+1}\}$ to G_k without two edges crossing (and G_{k+1} is planar). The addition of this new edge splits R into two regions. Consequently, in this case, $r_{k+1} = r_{k+1}$, $e_{k+1} = e_{k+1}$, and $v_{k+1} = v_k$. Thus, each side of the formula relating the number of regions, edges, and vertices increases by exactly one, so this formula is still true. In other words, $r_{k+1} = e_{k+1} - v_{k+1} + 2$. This case is illustrated in Figure 10(a).

In the second case, one of the two vertices of the new edge are not already in Gk. Suppose that ak+1 is in G_k but that b_{k+1} is not. Adding this new edge does not produce any new regions, because bk+1 must be in a region that has ak+1 on its boundary. Consequently, $r_{k+1} = r_k$. Moreover, $e_{k+1} = e_k + 1$ and $v_{k+1} = v_k + 1$. Each side of the formula relating the number of regions, edges, and vertices remains the same, so the formula is still true. In other words, $r_k+1 = e_k+1 - v_k+1 + 2$. This case is illustrated in Figure 10(b).

We have completed the induction argument. Hence, $r_n = e_n - v_n + 2$ for all n. Because the original graph is the graph G_e, obtained after e edges have been added, the theorem is true.

Euler's formula is illustrated in Example 4.

Euler's formula can be used to establish some inequalities that must be satisfied by planar graphs. One such inequality is given in Corollary 1.

Example 4 Suppose that a connected planar simple graph has 20 vertices, each of degree 3. Into how many regions does a representation of this planar graph split the plane? Solution: Thisgraphhas20vertices, eachofdegree3, sov = 20. Becausethesumofthedegrees of the vertices, 3v = 3·20 = 60, is equal to twice the number of edges, 2e, we have 2 e = 60, or e = 30. Consequently, from Euler's formula, the number of regions is r = e−v+2 = 30−20+2 = 12.

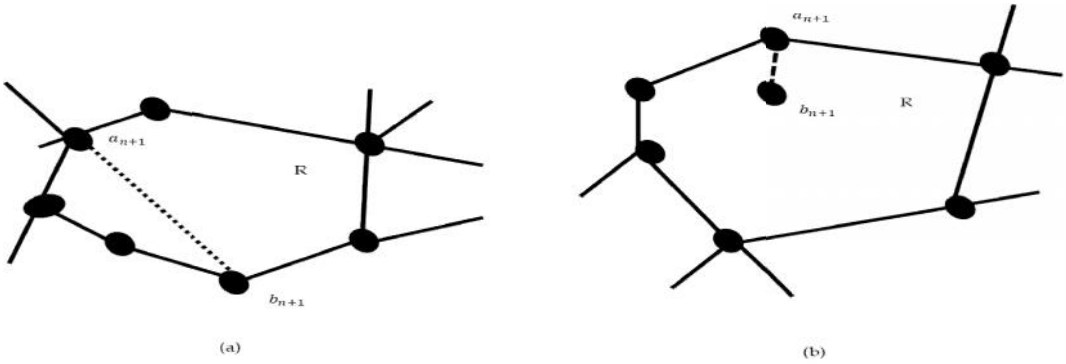


Figure 10: Adding an Edge to G_n to Produce G_{n+1}.

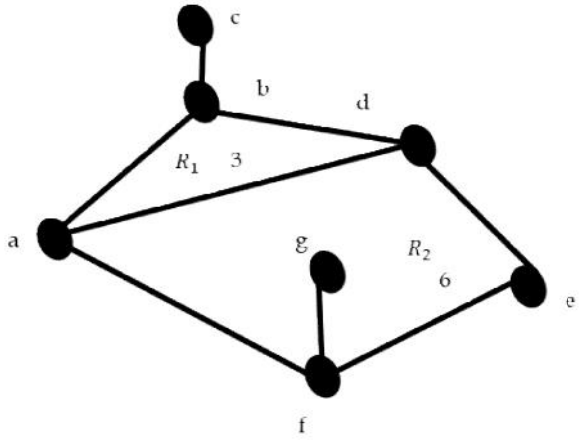


Figure 11: The Degrees of Regions.

COROLLARY 1 If G is a connected planar simple graph with e edges and v vertices, where v ≥ 3, then e ≤ 3v−6.

Before we prove Corollary 1, we will use it to prove the following useful result.

COROLLARY 2 If G is a connected planar simple graph, then G has a vertex of degree not exceeding five.

Proof: If G has one or two vertices, the result is true. If G has at least three vertices, by Corollary1 we know that e ≤ 3v−6, so 2e ≤ 6v−12. If the degree of every vertex were at least six, then because 2e =v∈V deg(v) (by the handshaking theorem), we would have 2e ≥ 6v. But this contradicts the inequality 2e ≤ 6v−12. It follows that there must be a vertex with degree no greater than five.

The proof of Corollary 1 is based on the concept of the degree of a region, which is defined to be the number of edges on the boundary of this region. When an edge occurs twice on the boundary (so that it is traced out twice when the boundary is traced out), it contributes two to the degree. We denote the degree of a region R by deg(R). The degrees of the regions of the graph shown in Figure 11 are displayed in the figure. The proof of Corollary 1 can now be given.

The proof of Corollary 1 is based on the concept of the degree of a region, which is defined to be the number of edges on the boundary of this region. When an edge occurs twice on the boundary (so that it is traced out twice when the boundary is traced out), it contributes two to the degree. We denote the degree of a region R by deg(R). The degrees of the regions of the graph shown in Figure 11 are displayed in the figure. The proof of Corollary 1 can now be given.

2e = ∑_(all regions R)▒ 〖deg(R) ≥ 3r〗

Hence, (2/3) e ≥ r.

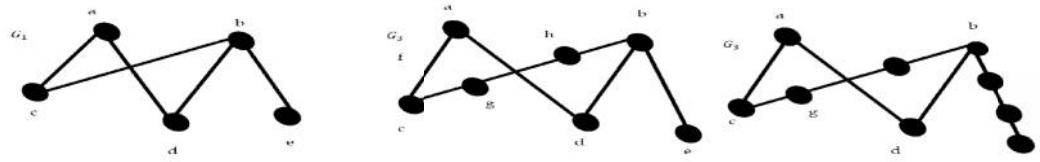


Figure 12 Homeomorphic Graphs.

Using r = e−v+2 (Euler's formula), we obtain e−v+2 ≤ (2/3) e. It follows that e/3 ≤ v−2. This shows that e ≤ 3v−6. This corollary can be used to demonstrate that K5 is nonplanar.

Example 5: Show that K_5 is nonplanar using Corollary 1. Solution: The graph K_5 has five vertices and 10 edges. However, the inequality e ≤ 3v−6 is not satisfied for this graph because e = 10 and 3v−6 = 9. Therefore, K_5 is not planar. ▲ Itwaspreviouslyshownthat$K_{3,3}$ isnotplanar. Note, however, thatthisgraphhassixvertices andnineedges. Thismeansthatthein equality = 9 ≤ 12 = 3 ·6−6issatisfied.Consequently,the fact that the inequality e ≤ 3v−6 is satisfied does not imply that a graph is planar. However, the following corollary of Theorem 1 can be used to show that $K_{3,3}$ is nonplanar COROLLARY 3 If a connected planar simple graph has e edges and v vertices with v ≥ 3 and no circuits of length three, then e ≤ 2v−4.

The proof of Corollary 3 is similar to that of Corollary 1, except that in this case the fact that there are no circuits of length three implies that the degree of a region must be at least four.

Example 6: Use Corollary 3 to show that $K_{3,3}$ is nonplanar.

Solution: Because $K_{3,3}$ has no circuits of length three (this is easy to see because it is bipartite), Corollary 3 can be used. $K_{3,3}$ has six vertices and nine edges. Because e = 9 and 2v−4 = 8, Corollary 3 shows that $K_{3,3}$ is nonplanar.

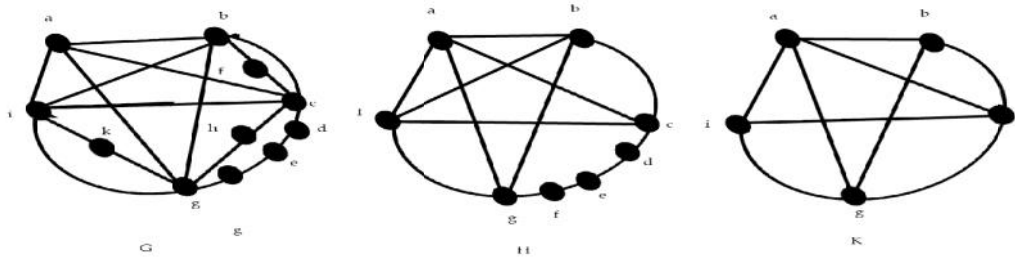


Figure 13: The Undirected Graph G, a Subgraph H Homeomorphic to K_5, and K_5.

Kuratowski's Theorem

We have seen that $K_{3,3}$ and K_5 are not planar. Clearly, a graph is not planar if it contains either of these two graphs as a subgraph. Surprisingly, all nonplanar graphs must contain a subgraph that can be obtained from $K_{3,3}$ or K_5 using certain permitted operations. If a graph is planar, so will be any graph obtained by removing an edge {u, v}and adding a new vertex w together with edges {u, w}and {w, v}. Such an operation is called an elementary subdivision. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called homeomorphic if they can be obtained from the same graph by a sequence of elementary subdivisions.

Example 7: Show that the graphs G_1, G_2, and G_3 displayed in Figure 12 are all homeomorphic.

Solution: These three graphs are homeomorphic because all three can be obtained from G_1 by elementary subdivisions. G_1 can be obtained from itself by an empty sequence of elementary subdivisions. To obtain G_2 from G_1 we can use this sequence of elementary subdivisions: (i) remove the edge {a,c}, add the vertex f, and add the edges{a,f} and {f,c};(ii) remove the edge{b,c}, add the vertex g, and add the edges{b,g}and{g,c}; and (iii) remove the edge {b,g},addthevertexh,andaddtheedges{g,h}and{b,h}.

The Polish mathematician Kazimierz Kuratowski established

Theorem 2in 1930, which characterizes planar graphs using the concept of graph homeomorphism.

Theorem 2 A graph is nonplanar if and only if it contains a subgraph homeomorphic to K3,3 or K5.

ItisclearthatagraphcontainingasubgraphhomeomorphictoK3,3 orK5 isnonplanar. However, theproofoftheconverse, namelythateverynonplanargraphcontainsasubgraphhomeomorphic to K3,3 or K5, is complicated and will not be given here.

Examples 8 and 9 illustrate how Kuratowski's theorem is used.

Example 8 Determine whether the graph G shown in Figure 13 is planar.

Solution: G has a subgraph H homeomorphic to K5. H is obtained by deleting h, j, and k and all edges incident with these vertices. H is homeomorphic to K5 because it can be obtained from K5 (with vertices a, b, c, g, and i) by a sequence of elementary subdivisions, adding the vertices d, e, and f. (The reader should construct such a sequence of elementary subdivisions.) Hence, G is nonplanar.

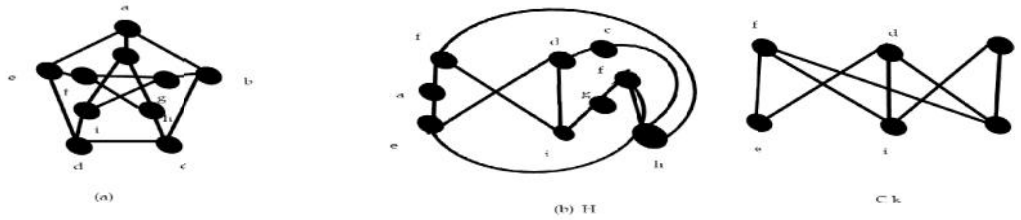


Figure 14 (a) The Petersen Graph, (b) a Subgraph H Homeomorphic to K3,3, and (c) K3,3.

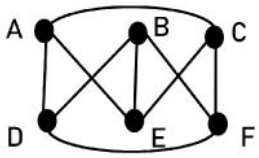
Example: IsthePetersengraph, showninFigure14(a), planar? (TheDanishmathematicianJuliusPetersen studied this graph in 1891; it is often used to illustrate various theoretical properties of graphs.)

Solution: The subgraph H of the Petersen graph obtained by deleting b and the three edges that have b as an endpoint, shown in Figure 14(b), is homeomorphic to K3,3, with vertex sets {f,d,j} and {e,i,h}, because it can be obtained by a sequence of elementary subdivisions, deleting {d,h} and adding {c,h} and {c,d}, deleting {e,f} and adding {a,e} and {a,f}, and deleting{i,j}and adding{g,i}and{g,j}. Hence, the Petersen graph is not planar.

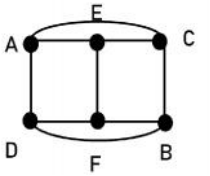
Summary

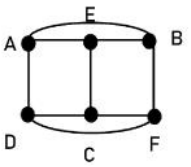
- We have learned what are Shortest-Path Problems
- We have learned what is a Shortest-Path Algorithm.
- We have learned what is Dijkstra's Algorithm.
- We have learned what are Planar Graphs
- We have learned how K3,3 is a non-planar Graph

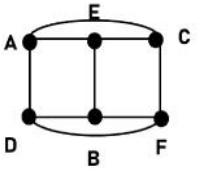
Self Assessment

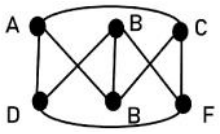


1. **A** planar representation of is.

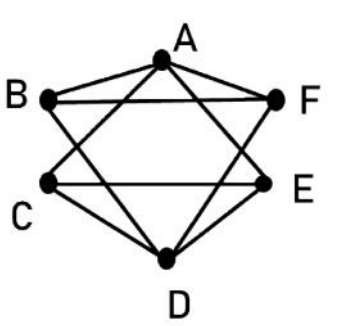
A. 

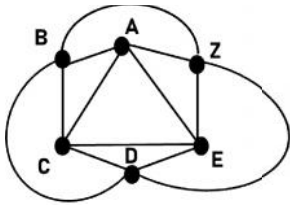
B. 

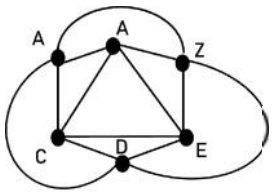
C. 

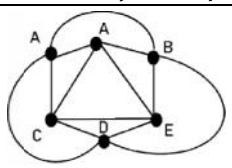
D. 

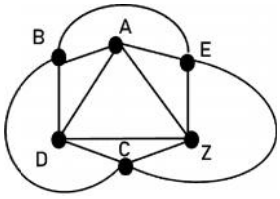
2. Draw a planer representation of each graph in fig below if possible.



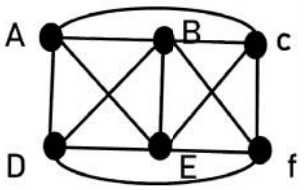
A. 

B. 

C. 

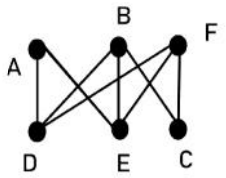
D. 

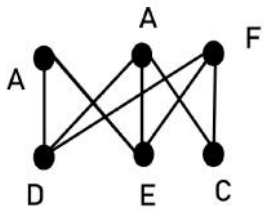
3. Draw a planer representation of each graph in fig below if possible

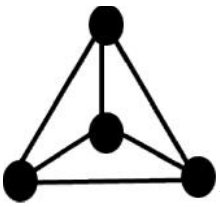


A. The graph has many planar representation

B. The graph is nonplaner.

C. 

D. 

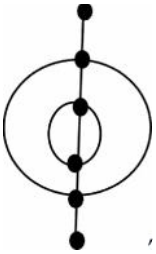
4. For the graph  which of the statement is true?

A. V=4, E=6, R=0.Hence using euler 's formula V-E+R=4-6+4=2.Also d=2

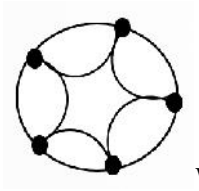
B. V=4, E=6, R=4.Hence using euler 's formula V-E+R=4-6+4=2.Also d=3

C. V=3, E=6, R=4.Hence using euler 's formula V-E+R=4-6+4=2.Also d=3

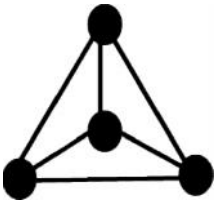
D. V=9, E=6, R=4.Hence using euler 's formula V-E+R=4-6+4=9.Also d=3

5. For the graph  , which of the statement is true?

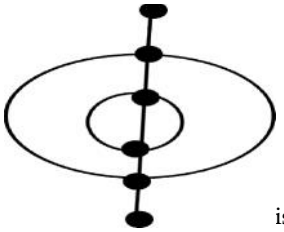
   A. V=6, E=0, R=5; SO, V-E+R=6-9+5=2.Here d=6 since two edges are counted twice.
   B. V=6, E=3, R=5; SO, V-E+R=6-9+5=12.Here d=6 since two edges are counted twice.
   C. V=6, E=9, R=5; SO, V-E+R=6-9+5=62.Here d=6 since two edges are counted twice.
   D. V=6, E=9, R=5; SO, V-E+R=6-9+5=2.Here d=16 since two edges are counted twice.

6. For the graph  which of the statement is true?

   A. V=4, E=10, R=7.Henc V-E+R 5-10 +7=2.Here d=52
   B. V=6, E=10, R=7.Henc V-E+R 5-10 +7=2.Here d=2
   C. V=7, E=10, R=7.Henc V-E+R 5-10 +7=2.Here d=4
   D. V=5, E=10, R=7.Henc V-E+R 5-10 +7=2.Here d=5

7. The minimum number n of colors required to paint  is

   A. n=4
   B. n=0
   C. n=8
   D. n=7

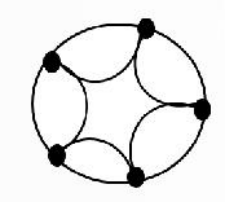
8. The minimum number n of colors required to paint  is

A. n=3

B. n=9

C. n=6

D. n=2

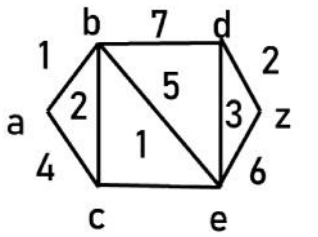
9. Find the minimum number n of colors required to paint 

A. only twelve colors are needed i.e, n=12

B. only two colors are needed i.e, n=2

C. only twenty two colors are needed i.e, n=22

D. only one colors are needed i.e, n=1

10. The shortest distance between sources a and destination z using Dijkstra 's algorithm for following graph 

is

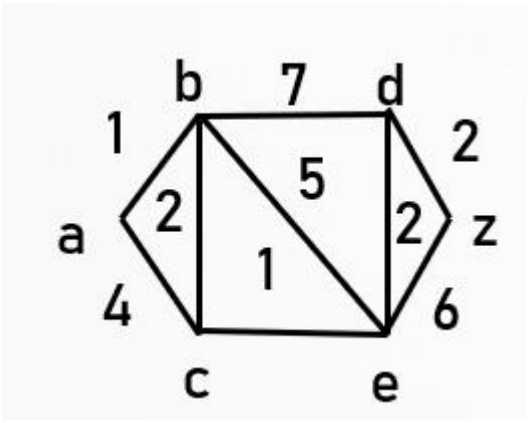
A. 9

B. 10

C. 11

D. 12

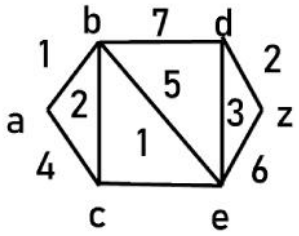
11. The shortest distance between sources a and destination z using Dijkstra 's algorithm for



is

A. 8
B. 10
C. 11
D. 12

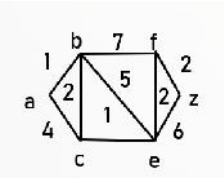
12. The shortest path between sources a and destination z using Dijkstra 's algorithm for following



graph is

A. Therefore, shortest path between a and z =(a-b-c-d-e-z)
B. Therefore, shortest path between a and z =(a-c-b-e-d-z)
C. Therefore, shortest path between a and z =(a-b-c-e-d-z)
D. Therefore, shortest path between a and z =(z-b-c-d-e-a)

The shortest path between sources a and destination z using Dijkstra 's algorithm

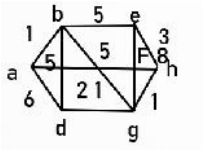


13. for following graph is

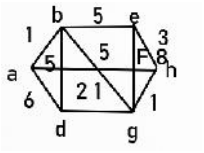
    A. Therefore, shortest path between a and z =(a-b-c-f-e-z)
    B. Therefore, shortest path between a and z =(a-c-b-e-f-z)
    C. Therefore, shortest path between a and z =(a-b-c-e-f-z)
    D. Therefore, shortest path between a and z =(z-b-c-f-e-a)

14. The shortest path between sources a and destination z using Dijkstra 's algorithm for following graph is

    A. Therefore, shortest path between a and z =(a-b-c-d-e-z)
    B. Therefore, shortest path between a and z =(a-c-b-e-d-z)
    C. Therefore, shortest path between a and z =(a-b-c-e-d-z)
    D. Therefore, shortest path between a and z =(z-b-c-d-e-a)



15. The shortest distance between sources a and destination z using Dijkstra 's algorithm for following graph is



    A. 8
    B. 9
    C. 11
    D. 12

Answers for Self Assessment

1.	C	2.	A	3.	B	4.	B	5.	B
6.	D	7.	A	8.	A	9.	B	10.	A
11.	A	12.	C	13.	C	14.	C	15.	B

Review Questions

Q1. What are some applications where it is necessary to findthe length of a longest simple path between two verticesin a weighted graph?

Q2. Show that a graph is planar if and only if it does not contain a subgraph that is a K_5 or $K_{3,3}$ configuration.

Q3. Prove that if G is a connected planar simple graph with e edges and v vertices where v >= 3, then e <= 3v - 6.

Q4. Is K_5 planar?

Q5. Is K_4 planar?

Further Readings

1. Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
2. Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.
3. Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.
4. Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 11: Graph Coloring

CONTENTS

Objective

11.1 GRAPH COLORINGS

11.2 Dual Maps and the Four Color Theorem

11.3 Applications of Graph Colorings

Summary

Self Assessment

Self Assessment Answer

Review Question

Further Reading

Objective

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- understand what is graph coloring.
- understand what is a 4-Color Map Theorem.
- understand what is a chromatic number

11.1 GRAPH COLORINGS

Problems related to the coloring of maps of regions, such as maps of parts of the world, have generated many results in graph theory. When a map∗ is colored, two regions with a common border are customarily assigned different colors. One way to ensure that two adjacent regions never have the same color is to use a different color for each region. However, this is inefficient, and on maps with many regions it would be hard to distinguish similar colors. Instead, a small number of colors should be used whenever possible. Consider the problem of determining the least number of colors that can be used to color a map so that adjacent regions never have the same color. For instance, for the map shown on the left in Figure 1, four colors suffice, but three colors are not enough. In the map on the right in Figure 1, three colors are sufficient (but two are not). Each map in the plane can be represented by a graph. To set up this correspondence, each region of the map is represented by a vertex. Edges connect two vertices if the regions represented by these vertices have a common border. Two regions that touch at only one point are not considered adjacent. The resulting graph is called the dual graph of the map. By the way in which dual graphs of maps are constructed, it is clear that any map in the plane has a planar dual graph. Figure 2 displays the dual graphs that correspond to the maps shown in Figure 1. The problem of coloring the regions of a map is equivalent to the problem of coloring the vertices of the dual graph so that no two adjacent vertices in this graph have the same color. We are now define a graph coloring.

DEFINITION 1 A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

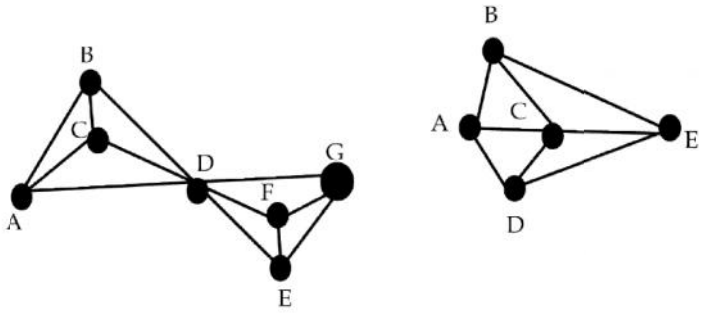


FIGURE 2 Dual Graphs of the Maps in Figure 1.

∗We will assume that all regions in a map are connected. This eliminates any problems presented by such geographical entities as Michigan.

A graph can be colored by assigning a different color to each of its vertices. However, for most graphs a coloring can be found that uses fewer colors than the number of vertices in the graph. What is the least number of colors necessary?

DEFINITION2. The chromatic number of a graph is the least number of colors needed for a coloring of this graph. The chromatic number of a graph G is denoted by $\chi(G)$. (Here χ is the Greek letter chi.)

Note that asking for the chromatic number of a planar graph is the same as asking for the minimum number of colors required to color a planar map so that no two adjacent regions are assigned the same color. This question has been studied for more than 100 years. The answer is provided by one of the most famous theorems in mathematics.

THEOREM 1 THE FOUR COLOR THEOREM The chromatic number of a planar graph is no greater than four.

The four-color theorem was originally posed as a conjecture in the 1850s. It was finally proved by the American mathematicians Kenneth Appel and Wolfgang Haken in 1976. Prior to 1976, many incorrect proofs were published, often with hard-to-find errors. In addition, many futile attempts were made to construct counterexamples by drawing maps that require more than four colors. (Proving the five color theorem is not that difficult; see Exercise 36.) Perhaps the most notorious fallacious proof in all of mathematics is the incorrect proof of the four color theorem published in 1879 by a London barrister and amateur mathematician, Alfred Kempe. Mathematicians accepted his proof as correct until 1890, when Percy Heawood found an error that made Kempe's argument incomplete. However, Kempe's line of reasoning turned out to be the basis of the successful proof given by Appel and Haken. Their proof relies on a careful case-by-case analysis carried out by computer. They showed that if the four color theorem were false, there would have to be a counterexample of one of approximately 2000 different types, and they then showed that none of these types exists. They used over 1000 hours of computer time in their proof. This proof generated a large amount of controversy, because computers played such an important role in it. For example, could there be an error in a computer program that led to incorrect results? Was their argument really a proof if it depended on what could be unreliable computer output? Since their proof appeared, simpler proofs that rely on checking fewer types of possible counterexamples have been found and a proof using an automated proof system has been created. However, no proof not relying on a computer has yet been found. Note that the four color theorem applies only to planar graphs. Non-planar graphs can have arbitrarily large chromatic numbers, as will be shown in Example above. Two things are required to show that the chromatic number of a graph is k. First, we must show that the graph can be colored with k colors. This can be done by constructing such a coloring. Second, we must show that the graph cannot be colored using fewer than k colors. Examples above illustrate how chromatic numbers can be found.

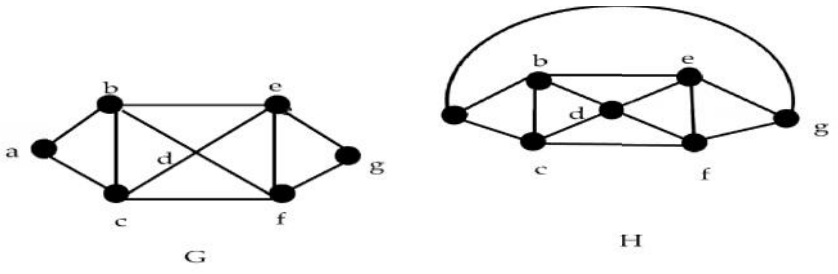


FIGURE 3The Simple Graphs G and H.

1. What are the chromatic numbers of the graphs G and H shown in Figure 3?

Solution: The chromatic number of G is at least three, because the vertices a, b, and c must be assigned different colors. To see if G can be colored with three colors assign, red to a, blue to b, and green to c Then, d can (and must) be colored red because it is adjacent to b and c. Furthermore, e can (and must) be colored green because it is adjacent only to vertices colored red and blue, and f can (and must) be colored blue because it is adjacent only to vertices colored red and green. Finally, g can (and must) be colored red because it is adjacent only to vertices colored blue and green. This produces a coloring of G using exactly three colors. Figure 4 displays such a coloring. The graph H is made up of the graph G with an edge connecting a and g. Any attempt to color H using three colors must follow the same reasoning as that used to color G, except at the last stage, when all vertices other than g have been colored. Then, because g is adjacent (in H) to vertices colored red, blue, and green, a fourth color, say brown, needs to be used. Hence, H has a chromatic number equal to 4. A coloring of H is shown in Figure 4.

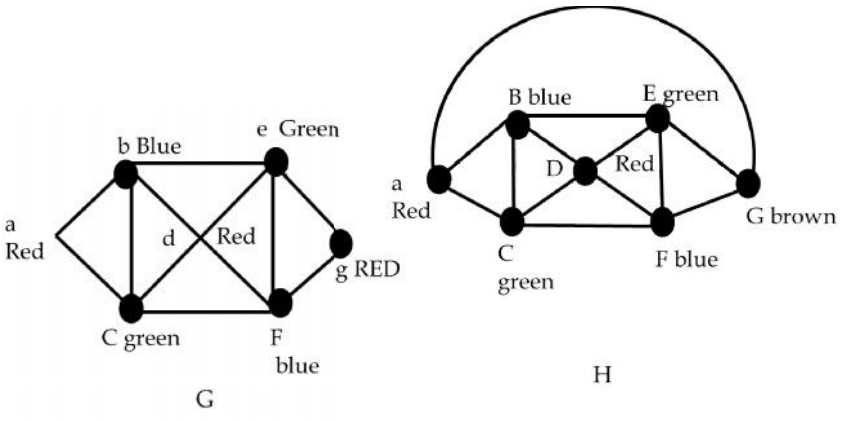
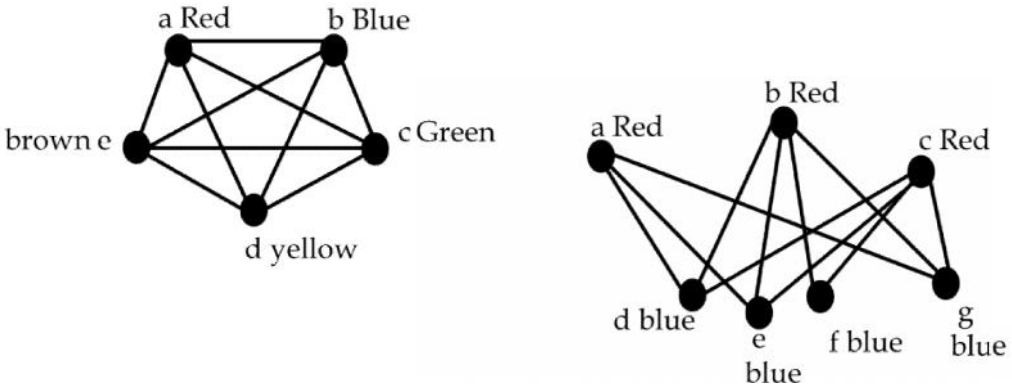


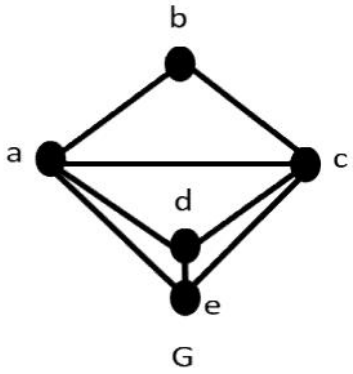
FIGURE 4 Colorings of the Graphs G and H.



Definition: Let G be a graph. The assignment of colours to the vertices of G, one colour to each vertex, so that the adjacent vertices are assigned different colours is called vertex colouring or colouring of the graph G. Definition: A graph G is n-colourable if there exists a colouring of G which uses n colours.

Definition: The minimum number of colours required to paint (colour) a graph G is called the chromatic number of G and is denoted by χ (G).

Find the chromatic number for the graph shown in the figure below:



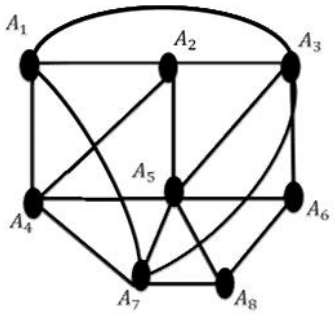
Solution: The triangle a b c needs three colours. Suppose that we assign colours c1, c2, c3 to a, b and c respectively. Since d is adjacent to a and c, d will have different colour than c1 and c3. So, we paint d by c2. Then e must be painted with a colour different from those of a, d and c, that is, we cannot colour e with c1, c2 or c3. Hence, we have to give e a fourth colour c4. Hence

$$\chi(G) = 4.$$

Consider a graph G. A vertex coloring or simply a coloring of G is an assignment of colors to the vertices of G such that adjacent vertices have different colors. We say that G is n-colorable if there exists a coloring of G which uses n colors. (Since the word "color" is used as a noun, we will try to avoid its use as a verb by saying, for example, "paint" G rather than "color" G when we are assigning colors to the vertices of G.) The minimum number of colors needed to paint G is called the chromatic number of G and is denoted by $\chi(G)$. Fig. 8-24 gives an algorithm by Welch and Powell for a coloring of a graph G. We emphasize that this algorithm does not always yield a minimal coloring of G.

Consider the graph G in Fig.below. We use the Welch-Powell Algorithm to obtain a coloring of G. Ordering the vertices according to decreasing degrees yields the following sequence:

A5, A 3, A 7, A 1, A 2, A 4, A 6, A8



The first color is assigned to vertices A5 and A1. The second color is assigned to vertices A3, A4, and A8. The third color is assigned to vertices A7, A2, and A6. All the vertices have been assigned a color, and so G is 3-colorable. Observe that G is not 2-colorable since vertices A1, A2, and A3, which are connected to each other, must be assigned different colors. Accordingly, $\chi(G) = 3$.

(b) Consider the complete graph Kn with n vertices. Since every vertex is adjacent to every other vertex, Kn requires n colors in any coloring. Thus $\chi(Kn) = n$.

There is no simple way to actually determine whether an arbitrary graph is n-colorable. However, the following theorem gives a simple characterization of 2-colorable graphs.

Theorem: The following are equivalent for a graph G:

(i) G is 2-colorable.

(ii) (ii) G is bipartite.

(iii) (iii) Every cycle of G has even length.

There is no limit on the number of colors that may be required for a coloring of an arbitrary graph since, for example, the complete graph Kn requires n colors. However, if we restrict ourselves to planar graphs, regardless of the number of vertices, five colors suffice. Specifically, in Problem we prove:

Theorem: Any planar graph is 5-colorable. Actually, since the 1850s mathematicians have conjectured that planar graphs are 4-colorable since every known planar graph is 4-colorable. Kenneth Appel and Wolfgang Haken finally proved this conjecture to be true in 1976. That is:

Four Color Theorem (Appel and Haken): Any planar graph is 4-colorable. We discuss this theorem in the next subsection.

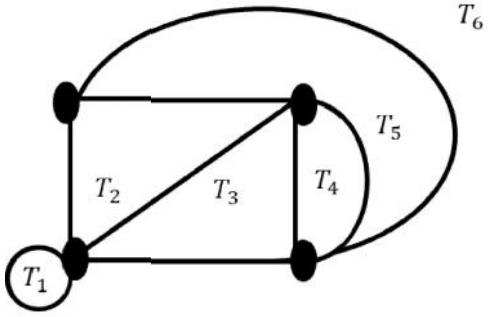
11.2 Dual Maps and the Four Color Theorem

Consider a map M, say the map M in Fig. below. In other words, M is a planar representation of a planar multigraph. Two regions of M are said to be adjacent if they have an edge in common. Thus, the regions r2 and r5 in Fig. below are adjacent, but the regions r3 and r5 are not. By a coloring of M, we mean an assignment of a color to each region of M such that adjacent regions have different colors. A map M is n-colorable if there exists a coloring of M which uses n colors. Thus, the map M in Fig. below is 3-colorable since the regions can be assigned the following colors:

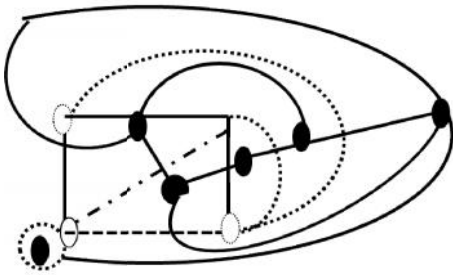
r1 red, r 2 white, r 3 red, r 4 white, r 5 red, r 6 blue

Observe the similarity between this discussion on coloring maps and the previous discussion on coloring graphs. In fact, using the concept of the dual map defined below, the coloring of a map can be shown to be equivalent to the vertex coloring of a planar graph.

Consider a map M. In each region of M, we choose a point, and if two regions have an edge in common then we connect the corresponding points with a curve through the common edge. These curves can be drawn so that they are none crossing. Thus we obtain a new map M∗, called the dual of M, such that each vertex of M∗ corresponds to exactly one region of M.Figure(b) below showsthedualofthemapofFig.below.One can prove that each region of M∗ will contain exactly one vertex of M and that each edge of M∗ will intersect exactly one edge of M and vice versa. Thus, M will be the dual of the map M∗.



(a)



(b)

Observe that any coloring of the regions of a map M will correspond to a coloring of the vertices of the dual map M∗. Thus, M is n-colorable if and only if the planar graph of the dual map M∗ is

vertex n-colorable. Thus, the above theorem can be restated as follows: Four Color Theorem (Appel and Haken): If the regions of any map M are colored so that adjacent regions have different colors, then no more than four colors are required.

The proof of the above theorem uses computers in an essential way. Specifically, Appel and Haken first showed that if the four-color theorem was false, then there must be a counterexample among one of approximately 2000 different types of planar graphs. They then showed, using the computer, that none of these types of graphs has such a counter example. The examination of each different type of graph seems to be beyond the grasp of human beings without the use of a computer. Thus, the proof, un like most proofs in mathematics, is technology dependent; that is, it depended on the development of high-speed computers.

2. What is the chromatic number of Kn?

Solution: A coloring of Kn can be constructed using n colors by assigning a different color to each vertex. Is there a coloring using fewer colors? The answer is no. No two vertices can be assigned the same color, because every two vertices of this graph are adjacent. Hence, the chromatic number of Kn is n. That is, $\chi(Kn) = n$. (Recall that Kn is not planar when $n \geq 5$, so this result does not contradict the four-color theorem.) A coloring of K5 using five colors is shown in Figure 5. ▲

3. What is the chromatic number of the complete bipartite graph Km, n, where m and n are positive integers?

Solution: The number of colors needed may seem to depend on m and n. However, as Theorem, only two colors are needed, because Km, n is a bipartite graph. Hence, χ (Km, n) = 2. This means that we can color the set of m vertices with one color and the set of n vertices with a second color. Because edges connect only a vertex from the set of m vertices and a vertex from the set of n vertices, no two adjacent vertices have the same color. A coloring of K 3,4 with two colors is displayed in Figure 6.

4. What is the chromatic number of the graph Cn, where $n \geq 3$? (Recall that Cn is the cycle with n vertices.)

Solution: We will first consider some individual cases. To begin, let n = 6. Pick a vertex and color its red. Proceed clockwise in the planar depiction of C6 shown in Figure 7. It is necessary to assign a second color, say blue, to the next vertex reached. Continue in the clockwise direction; the third vertex can be colored red, the fourth vertex blue, and the fifth vertex red. Finally, the sixth vertex, which is adjacent to the first, can be colored blue. Hence, the chromatic number of C6 is 2. Figure 7 displays the coloring constructed here.

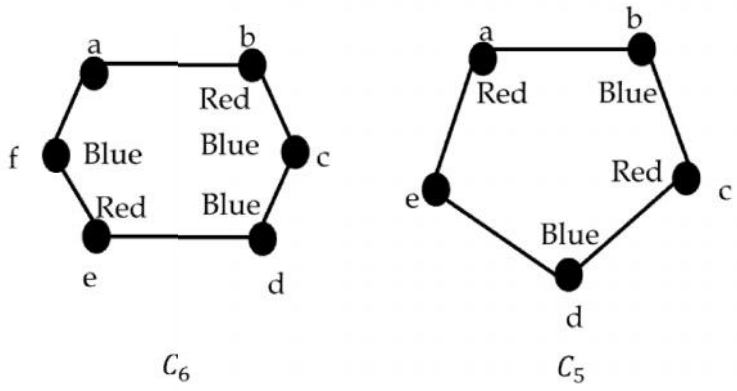


FIGURE 7 Colorings of C_5 and C_6.

Next, let n = 5 and consider C5. Pick a vertex and color its red. Proceeding clockwise, it is necessary to assign a second color, say blue, to the next vertex reached. Continuing in the clockwise direction, the third vertex can be colored red, and the fourth vertex can be colored blue. The fifth vertex cannot be colored either red or blue, because it is adjacent to the fourth vertex and the first vertex. Consequently, a third color is required for this vertex. Note that we would have also needed three colors if we had colored vertices in the counterclockwise direction. Thus, the chromatic number of

C5 is 3. A coloring of C5 using three colors is displayed in Figure 7. In general, two colors are needed to color Cn when n is even. To construct such a coloring, simply pick a vertex and color its red. Proceed around the graph in a clockwise direction (using a planar representation of the graph) coloring the second vertex blue, the third vertex red, and so on. The nth vertex can be colored blue, because the two vertices adjacent to it, namely the(n−1)st and the first vertices, are both colored red. When n is odd and n>1, the chromatic number of Cn is 3. To see this, pick an initial vertex. To use only two colors, it is necessary to alternate colors as the graph is traversed in a clockwise direction. However, the nth vertex reached is adjacent to two vertices of different colors, namely, the first and (n−1)st. Hence, a third color must be used. We have shown that $\chi(Cn) = 2$ if n is an even positive integer with $n \geq 4$ and $\chi(Cn) = 3$ if n is an odd positive integer with $n \geq 3$.

The best algorithms known for finding the chromatic number of a graph have exponential worst-case time complexity (in the number of vertices of the graph).Even the problem of finding an approximation to the chromatic number of a graph is difficult. It has been shown that if there were an algorithm with polynomial worst-case time complexity that could approximate the chromatic number of a graph up to a factor of 2 (that is, construct a bound that was no more than double the chromatic number of the graph), then an algorithm with polynomial worst-case time complexity for finding the chromatic number of the graph would also exist.

11.3 Applications of Graph Colorings

Graph coloring has a variety of applications to problems involving scheduling and assignments. (Note that because no efficient algorithm is known for graph coloring, this does not lead to efficient algorithms for scheduling and assignments.) Examples of such applications will be given here. The first application deals with the scheduling of final exams.

5. Scheduling Final Exams How can the final exams at a university be scheduled so that no student has two exams at the same time?

Solution: This scheduling problem can be solved using a graph model, with vertices representing courses and with an edge between two vertices if there is a common student in the courses they represent. Each time slot for a final exam is represented by a different color. A scheduling of the exams corresponds to a coloring of the associated graph.

For instance, suppose there are seven finals to be scheduled. Suppose the courses are numbered 1 through 7. Suppose that the following pairs of courses have common students: 1 and 2, 1 and 3, 1 and 4, 1 and 7, 2 and 3, 2 and 4, 2 and 5, 2 and 7, 3 and 4, 3 and 6, 3 and 7, 4 and 5, 4 and 6, 5 and 6, 5 and 7, and 6 and 7. In Figure 8 the graph associated with this set of classes is shown. A scheduling consists of a coloring of this graph.

Because the chromatic number of this graph is 4, four time slots are needed. According of the graph using four colors and the associated schedule are shown in Figure 9.

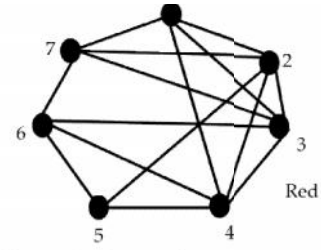


Figure 9: the graph representing the scheduling of final exams

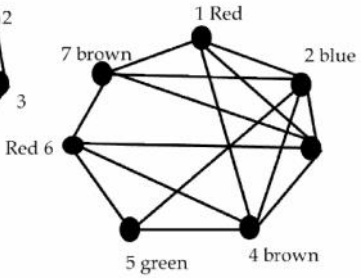


Figure 9: using coloring to schedule of final exams

Now consider an application to the assignment of television channels.

6. Frequency Assignments Television channels 2 through 13 are assigned to stations in North America so that no two stations within 150 miles can operate on the same

channel. How can the assignment of channels be modeled by graph coloring?

Solution: Construct a graph by assigning a vertex to each station. Two vertices are connected by anedgeiftheyarelocatedwithin150milesofeachother.Anassignmentofchannelscorresponds to a coloring of the graph, where each color represents a different channel. ▲

An application of graph coloring to compilers is considered in Example 7.

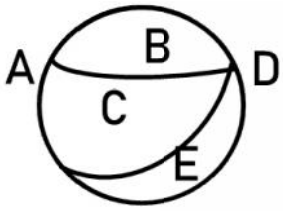
7. Index Registers In efficient compilers the execution of loops is speeded up when frequently used variables are stored temporarily in index registers in the central processing unit, instead of in regular memory. For a given loop, how many index registers are needed? This problem can be addressed using a graph coloring model. To set up the model, let each vertex of a graph represent a variable in the loop. There is an edge between two vertices if the variables they represent must be stored in index registers at the same time during the execution of the loop. Thus, the chromatic number of the graph gives the number of index registers needed, because different registers must be assigned to variables when the vertices representing these variables are adjacent in the graph.

Summary

- We have learned what is graph coloring.
- We have learned what is a 4-Color Map Theorem.
- We have learned what is a chromatic number
-

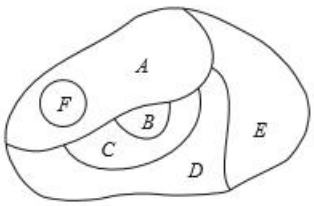
Self Assessment

1. The number of colors needed to color the map so that no two adjacent regions have the same color is.



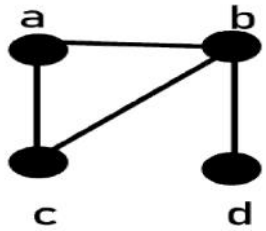
A. Four colors
B. Three colors
C. Two colors
D. Five colors

2. The number of colors needed to color the map so that no two adjacent regions have the same coloris



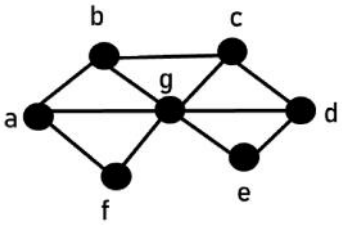
A. Three color
B. Two colors
C. One colors
D. None of these

3. The chromatic number of the given graph is



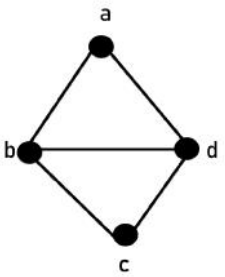
A. 2
B. 7
C. 1
D. 3

4. The number of the given graphis



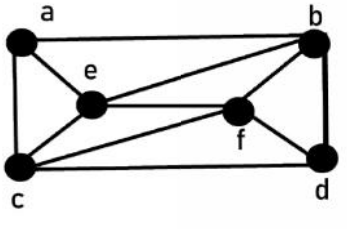
A. 1
B. 2
C. 3
D. 4

5. The chromatic number of the given graphis



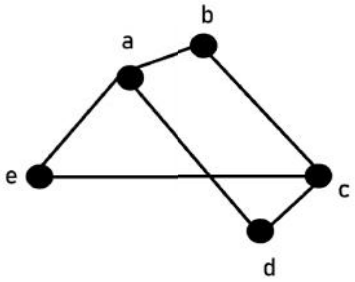
A. 4
B. 3
C. 2
D. 1

6. The chromatic number of the given graphis



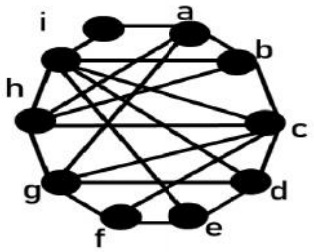
A. 1
B. 2
C. 3
D. 4

7. The chromatic number of the given graphis



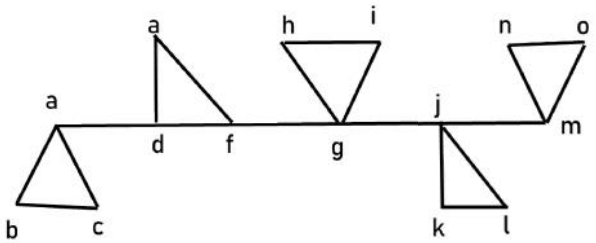
A. 1
B. 3
C. 2
D. 4

8. The chromatic number of the given graphis



A. 1
B. 2
C. 3
D. 4

9. The chromatic number of the given graph is



A. 8
B. 3
C. 0
D. 6

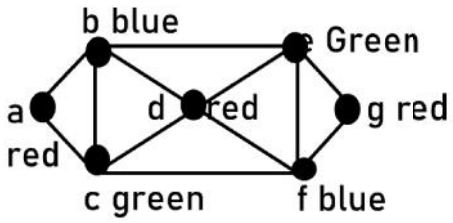
10. Which graphs have a chromatic number of 1?

A. Graphs with no edges
B. Graphs with 2 edges
C. Graphs with 3 edges
D. Graphs with 4 edges

11. What is the chromatic number of W_n?

A. 0 if n is even, 3 if n is odd
B. 1 if n is even, 5 if n is odd
C. 2 if n is even, 2 if n is odd
D. 3 if n is even, 4 if n is odd

12. The chromatic number of the graphis



A. 1
B. 2
C. 3
D. 4

13. What is the chromatic number of K_n?

A. n
B. n+1
C. n-1
D. n-2

14. The chromatic number of the complete bipartite graph $K_{m, n}$, where m and n are positive integers is

A. 2
B. 3
C. 4
D. 5

15. What is the chromatic number of the graph Cn, where n ≥ 3? (Recall that Cn is the cycle with n vertices.)

A. 2
B. 3
C. 4
D. 5

Self Assessment Answer

1.	A	2.	A	3.	D	4.	C	5.	B
6.	C	7.	C	8.	D	9.	B	10.	A
11.	D	12.	C	13.	A	14.	A	15.	A

Review Question

1. Schedule the final exams for Math 115, Math 116, Math 185, Math 195, CS 101, CS 102, CS 273, and CS 473, using the fewest number of different time slots, if there are no students taking both Math 115 and CS 473, both Math 116 and CS 473, both Math 195 and CS 101, both Math 195 and CS 102, both Math 115 and Math 116, both Math 115 and Math 185, and both Math 185 and Math 195, but there are students in every other pair ofcourses.
2. The mathematics department has six committees, each meeting once a month. Howmany different meeting times must be used to ensure that no member is scheduled to attend two meetings at the same time if the committees are C1 = {Arlinghaus, Brand, Zaslavsky}, C2 = {Brand,Lee, Rosen}, C3 = {Arlinghaus, Rosen, Zaslavsky}, C4 = {Lee, Rosen, Zaslavsky}, C5 = {Arlinghaus, Brand}, and C6 = {Brand, Rosen, Zaslavsky}?
3. What is the chromatic number of Wn?
4. Show that a simple graph that has a circuit with an odd number of vertices in it cannot be colored using two colors.

5.What is the least number of colors needed to color a map of the United States? Do not consider adjacent states that meet only at a corner. Suppose that Michigan is one region. Consider the vertices representing Alaska andHawaii as isolated vertices.

Further Reading

- Rosen, Kenneth H. "Discrete Mathematics and Its Applications."
- Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.
- Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.
- Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 12: Trees

CONTENTS

Objectives

Introduction

12.1 Rooted Trees

12.2 ORDERED ROOTEDTREES

12.3 Properties of Trees

Summary

Self Assessment

Answer for Self Assessment

Review Question

Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- Understand what are Trees
- Understand what different parts of trees

Introduction

Trees: A connected graph that contains no simple circuits is called a tree. Trees were used as long ago as 1857, when the English mathematician Arthur Cayley used them to count certain types of chemical compounds. Since that time, trees have been employed to solve problems in a wide variety of disciplines, as the examples in this unit will show. Trees are particularly useful in computer science, where they are employed in a wide range of algorithms. For instance, trees are used to construct efficient algorithms for locating items in a list. They can be used in algorithms, such as Huffman coding, that construct efficient codes saving costs in data transmission and storage. Trees can be used to study games such as checkers and chess and can help determine winning strategies for playing these games. Trees can be used to model procedures carried out using a sequence of decisions. Constructing these models can help determine the computational complexity of algorithms based on a sequence of decisions, such as sorting algorithms. Proceduresforbuildingtreescontainingeveryvertexofagraph, includingdepth-firstsearch and breadth-first search, can be used to systematically explore the vertices of a graph. Exploring the vertices of a graph via depth-first search, also known as backtracking, allows for the systematic search for solutions to a wide variety of problems, such as determining how eight queens can be placed on a chessboard so that no queen can attack another. We can assign weights to the edges of a tree to model many problems. For example, using weighted trees we can develop algorithms to construct networks containing the least expensive set of telephone lines linking different network nodes.

In previous unit we showed how graphs can be used to model and solve many problems. In this unit we will focus on a particular type of graph called a tree, so named because such graphs resemble trees. For example, family trees are graphs that represent genealogical charts. Family trees use vertices to represent the members of a family and edges to represent parent– child relationships. The family tree of the male members of the Bernoulli family of Swiss mathematicians is shown in Figure 1. The undirected graph representing a family tree (restricted to people of just one gender and with no inbreeding) is an example of a tree.

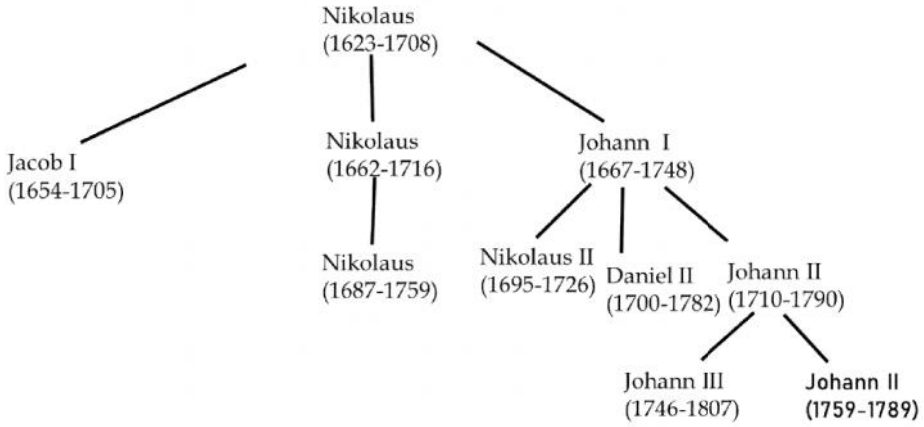


FIGURE 1 The Bernoulli Family of Mathematicians.

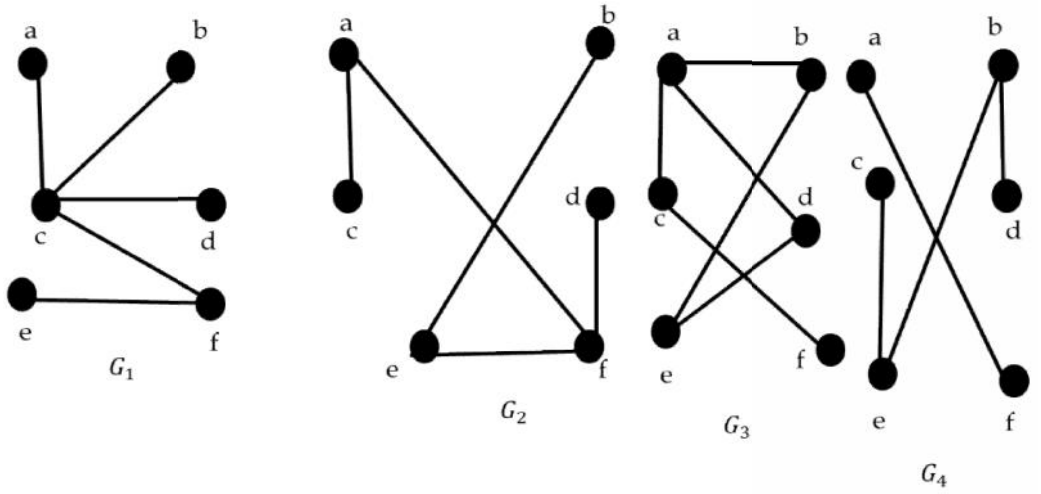


FIGURE 2 Examples of Trees and Graphs That Are Not Trees.

DEFINITION 1 A tree is a connected undirected graph with no simple circuits.

Because a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops. Therefore, any tree must be a simple graph. EXAMPLE 1. Which of the graphs shown in Figure 2 are trees?

Solution: G_1 and G_2 are trees, because both are connected graphs with no simple circuits. G_3 is not a tree because e, b, a, d, e is a simple circuit in this graph. Finally, G_4 is not a tree because it is not connected. ▲

Anyconnectedgraphthatcontainsnosimplecircuitsisatree.Whataboutgraphscontaining no simple circuits that are not necessarily connected? These graphs are called forests and have the property that each of their connected components is a tree. Figure 3 displays a forest. Trees are often defined as undirected graphs with the property that there is a unique simple pathbetweeneverypairofvertices. Theorem1showsthatthisalternativedefinitionisequivalent to our definition.

THEOREM 1 An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

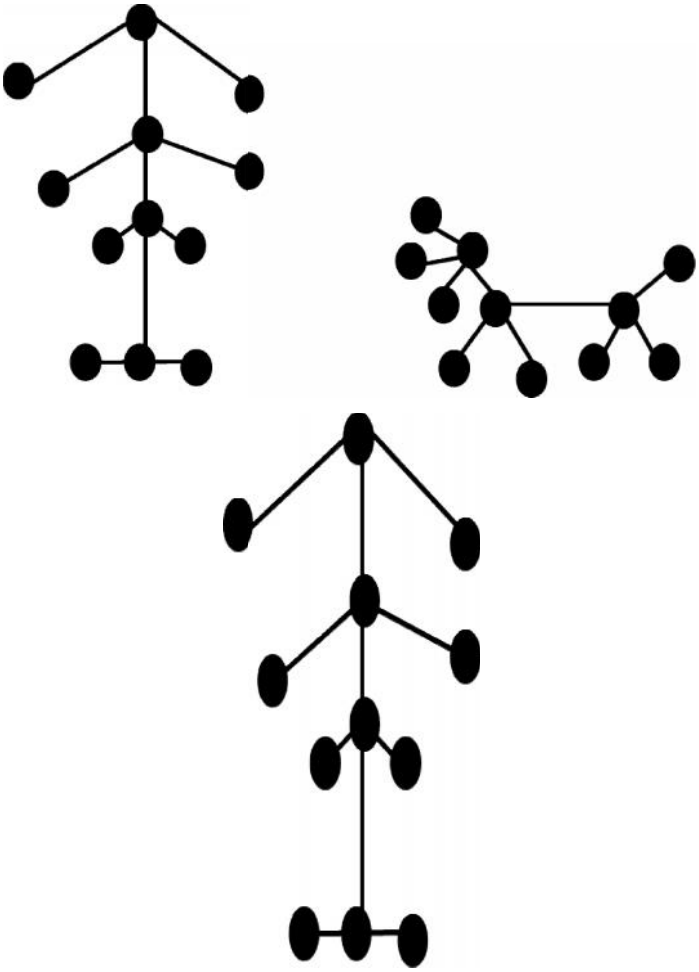


FIGURE 3 Example of a Forest.

Proof: First assume that T is a tree. Then T is a connected graph with no simple circuits. Let x and y be two vertices of T. Because T is connected, there is a simple path between x and y. Moreover, this path must be unique, for if there were a second such path, the path formed by combining the first path from x to y followed by the path from y to x obtained by reversing the order of the second path from x to y would form a circuit. This impliesthat there is a simple circuit in T. Hence, there is a unique simple path between any two vertices of a tree. Now assume that there is a unique simple path between any two vertices of a graph T. Then T is connected, because there is a path between any two of its vertices. Furthermore, T can have no simple circuits. To see that this is true suppose, T had a simple circuit that contained the vertices x and y,Then there would be two simple paths between x and y, because the simple circuit is made up of a simple path from x to y and a second simple path from y to x. Hence, a graph with a unique simple path between any two vertices is a tree.

12.1 Rooted Trees

In many applications of trees, a particular vertex of a tree is designated as the root. Once we specify a root, we can assign a direction to each edge as follows. Because there is a unique path from the root to each vertex of the graph (by Theorem 1), we direct each edge away from the root. Thus, a tree together with its root produces a directed graph called a rooted tree.

DEFINITION 2 A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

Rooted trees can also be defined recursively.. We can change an unrooted tree into a rooted tree by choosing any vertex as the root. Note that different choices of the root produce different rooted trees. For instance, Figure 4 displays the rooted trees formed by designating a to be the root and c to be the root, respectively, in the tree T. We usually draw a rooted tree with its root at the top of the graph. The arrows indicating the directions of the edges in a rooted tree can be omitted, because the

choice of root determines the directions of the edges. The terminology for trees has botanical and genealogical origins. Suppose that T is a rooted tree. If v is a vertex in T other than the root, the parent of v is the unique vertex u such that there is a directed edge from u to v (the reader should show that such a vertex is unique). When u is the parent of v, v is called a child of u. Vertices with the same parent are called siblings. The ancestors of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root (that is, its parent, its parent's parent, and so on, until the root is reached). The descendants of a vertex v are those vertices that have v as

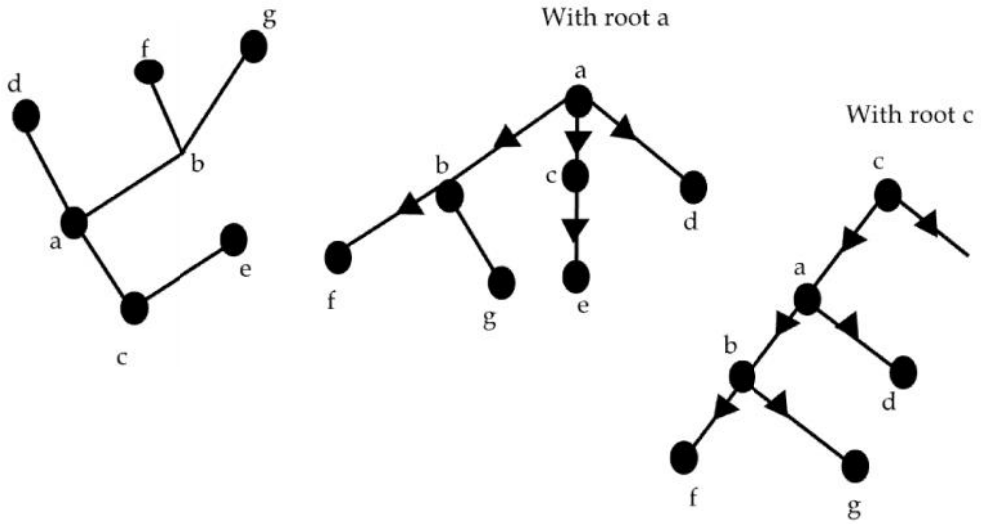


FIGURE 4 A Tree and Rooted Trees Formed by Designating Two Different Roots.

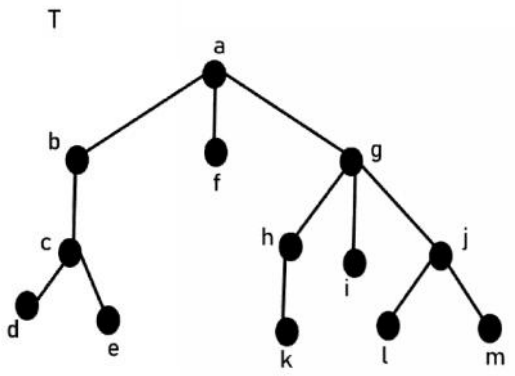


Figure 5 A Rooted tree T

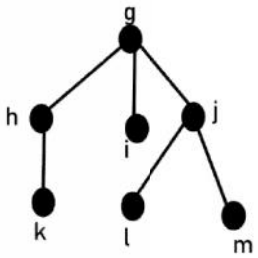


Figure 6 the subtree rooted at g

an ancestor. A vertex of a rooted tree is called a leaf if it has no children. Vertices that have children are called internal vertices. The root is an internal vertex unless it is the only vertex in the graph, in which case it is a leaf. If a is a vertex in a tree, the subtree with a as its root is the subgraph of the tree consisting of a and its descendants and all edges incident to these descendants.

In the rooted tree T (with root a) shown in Figure 5, find the parent of c, the children of g, the siblings of h, all ancestors of e, all descendants of b, all internal vertices, and all leaves. What is the subtree rooted at g?

Solution: The parent of c is b. The children of g are h, i, and j. The siblings of h are i and j. The ancestors of e are c, b, and a. The descendants of b are c, d, and e. The internal vertices are a, b, c, g, h, and j. The leaves are d, e, f, i, k, l, and m. The subtree rooted at g is shown in Figure 6. ▲

Rooted trees with the property that all of their internal vertices have the same number of children are used in many different applications. Later we will use such trees to study problems involving searching, sorting, and coding.

DEFINITION 3 A rooted tree is called an m-ary tree if every internal vertex has no more than m children. The tree is called a full m-ary tree if every internal vertex has exactly m children. An m-ary tree with m = 2 is called a binary tree.

Are the rooted trees in Figure 7 full m-ary trees for some positive integer m?

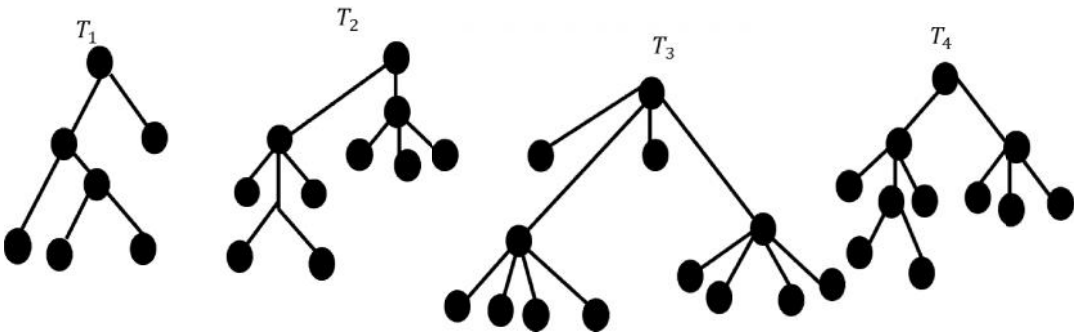
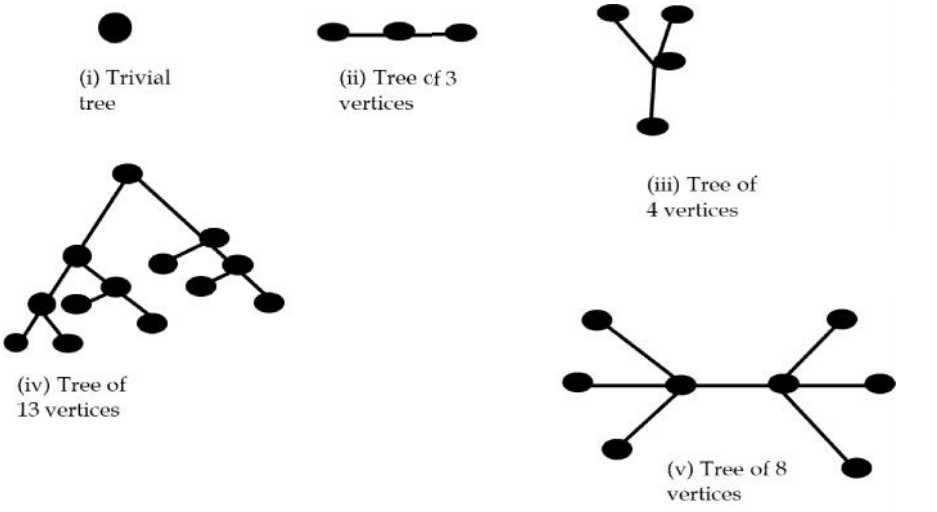


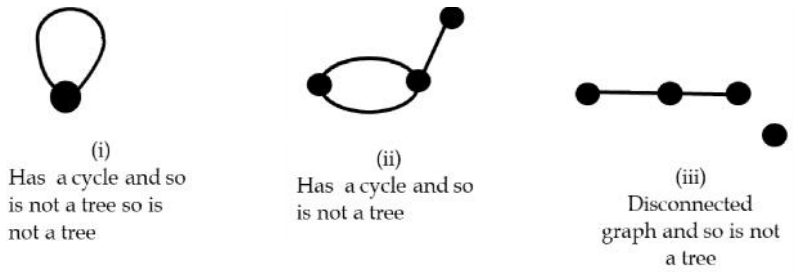
FIGURE 7 Four Rooted Trees.

Solution: T_1 is a full binary tree because each of its internal vertices has two children. T_2 is a full 3-ary tree because each of its internal vertices has three children. In T_3 each internal vertex has five children, so T_3 is a full 5-ary tree. T4 is not a full m-ary tree for any m because some of its internal vertices have two children and others have three children. ▲

Definition: A graph is said to be a Tree if it is a connected acyclic graph. A trivial tree is a graph that consists of a single vertex. An empty tree is a tree that does not have any vertices or edges.

For example, the graphs shown below are all trees.

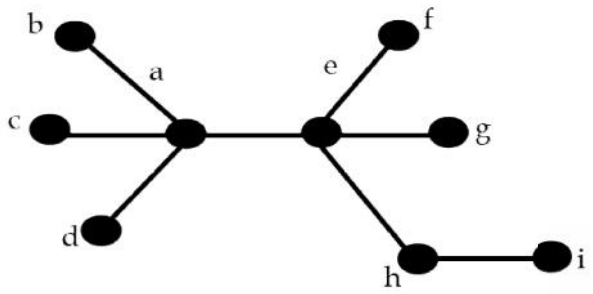




Definition: A collection of disjoint trees is called a forest.

Thus, a graph is a forest if and only if it is circuit free. Definition: A vertex of degree 1 in a tree is called a leaf or a terminal node or a terminal vertex. Definition: A vertex of degree greater than 1 in a tree is called a Branch node or Internal node or Internal vertex.

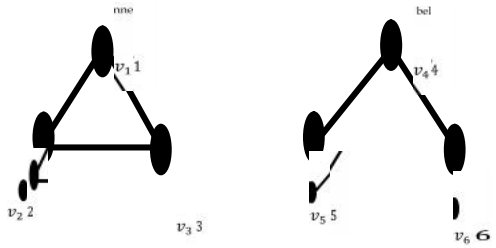
Consider the tree shown below:



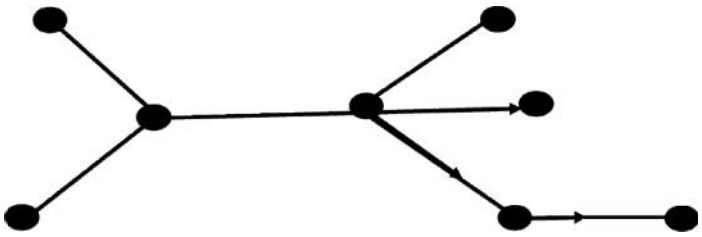
In this tree the vertices b, c, d, f, g, and i are leaves whereas the vertices a, e, h are branch nodes.

Construct a graph that has 6 vertices and 5 edges but is not a tree.

Solution: We have, No. of vertices = 6, No. of edges = 5. So, the condition e = v – 1 is satisfied. Therefore, to construct graph with six vertices and 5 edges that is not a tree, we should keep in mind that the graph should not be connected. The graph shown below has 6 vertices and 5 edges but is not connected.



Definition: A directed graph is said to be a directed tree if it becomes a tree when the direction of edges is ignored. For example, the graph shown below is a directed tree.



Definition: A directed tree is called a rooted tree if there is exactly one vertex whose incoming degree is 0 and the incoming degrees of all other vertices are 1.

The vertex with incoming degree 0 is called the root of the rooted tree. A tree T with root v0 will be denoted by (T, v_0).

Definition: In a rooted tree, a vertex, whose outgoing degree is 0 is called a leaf or terminal node, whereas a vertex whose outgoing degree is non - zero is called a branch node or an internal node.

Definition: Let u be a branch node in a rooted tree. Then a vertex v is said to be child (son or offspring) of u if there is an edge from u to v. In this case u is called parent (father) of v.

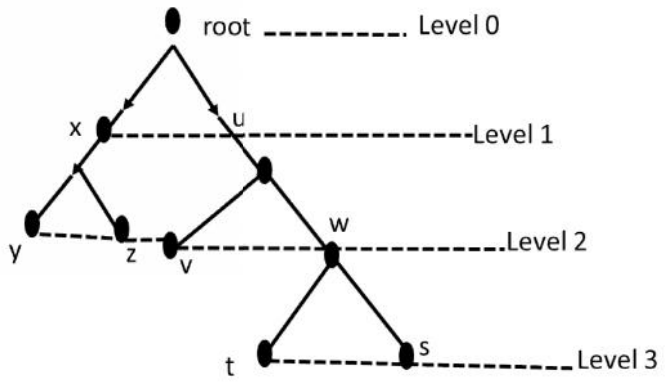
Definition: Two vertices in a rooted tree are said to be siblings (brothers) if they are both children of same parent.

Definition: A vertex v is said to be a descendent of a vertex u if there is a unique directed path from u to v. In this case u is called the ancestor of v.

Definition: The level (or path length) of a vertex u in a rooted tree is the number of edges along the unique path between u and the root.

Definition: The height of a rooted tree is the maximum level to any vertex of the tree.

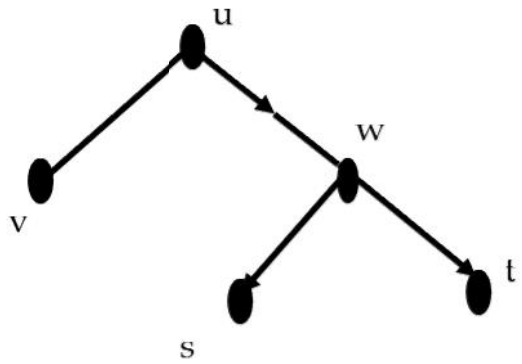
As an example of these terms consider the rooted tree shown below:



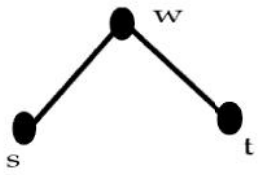
Here y is a child of x; x is the parent of y and z. Thus, y and z are siblings. The descendents of u are v, w, t and s. Levels of vertices are shown in the figure. The height of this rooted tree is 3.

Definition: Let u be a branch node in the tree T = (V, E). Then the subgraph T′ = (V′, E′) of T such that the vertices set V′ contains u and all of its descendents and E′ contains all the edges in all directed paths emerging from u is called a subtree with u as the root.

Definition: Let u be a branch node. By a subtree of u, we mean a subtree that has child of u as root. In the above example, we note that the figure shown below is a subtree of T,



Where as the figure shown below is a subtree of the branch node u.



is a subtree of the branch node u.

Let

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$

and let E =

$$(\{v_2, v_1), (v_2, v_3), (v_4, v_2), (v_4, v_5), (v_4, v_6), (v_6, v_7), (v_5, v_8)\}.$$

Show that (V, E) is rooted tree. Identify the root of this tree.

Solution: We note that

Incoming degree of $v_1 = 1$

Incoming degree of $v_2 = 1$

Incoming degree of $v_3 = 1$

Incoming degree of $v_4 = 0$

Incoming degree of $v_5 = 1$

Incoming degree of $v_6 = 1$

Incoming degree of $v_7 = 1$

Incoming degree of $v_8 = 1$

Since incoming degree of the vertex v_4 is 0, it follows that v_4 is root.

Further,

Outgoing degree of $v_1 = 0$

Outgoing degree of $v_3 = 0$

Outgoing degree of $v_7 = 0$

Outgoing degree of $v_8 = 0$

Therefore v_1, v_2, v_7, v_8 are leaves.

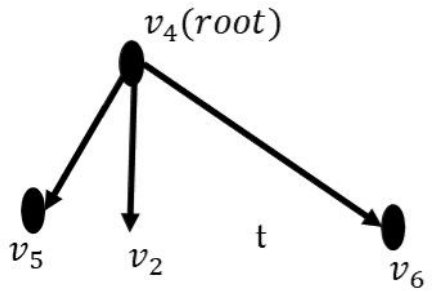
Also, Outgoing degree of $v_2 = 2$

Outgoing degree of $v_4 = 3$

Outgoing degree of $v_5 = 1$

Outgoing degree of $v_6 = 1$

Now the root v_4 is connected to v_2, v_5 and v_6. So, we have



Now v_2 is connected to v_1 and v_3, v_5 is connected to v_8, v_6 is connected to v_7. Thus, we have

We thus have a connected acyclic graph and so (V, E) is a rooted tree with root v_4.

Definition: A rooted tree in which the edges incident from each branch node are labeled with integers 1, 2, 3, is called an ordered tree.

12.2 Ordered Rootedtrees

An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered. Ordered rooted trees are drawn so that the children of each internal vertex are shown in order from left to right. Note that a representation of a rooted tree in the conventional way determines an ordering for its edges. We will use such orderings of edges in drawings without explicitly mentioning that we are considering a rooted tree to be ordered. In an ordered binary tree (usually called just a binary tree), if an internal vertex has two children, the first child is called the left child and the second child is called the right child, The tree rooted at the left child of a vertex is called the left subtree of this vertex, and the tree rooted at the right child of a vertex is called the right subtree of the vertex. The reader should note that for some applications every vertex of a binary tree, other than the root, is designated as a right or a left child of its parent. This is done even when some vertices have only one child. We will make such designations whenever it is necessary, but not otherwise. Ordered rooted trees can be defined recursively.

What are the left and right children of d in the binary tree T shown in Figure 8(a) (where the order is that implied by the drawing)? What are the left and right subtrees of c?

Solution: The left child of d is f and the right child is g. We show the left and right subtrees of c in Figures 8(b) and 8(c), respectively.

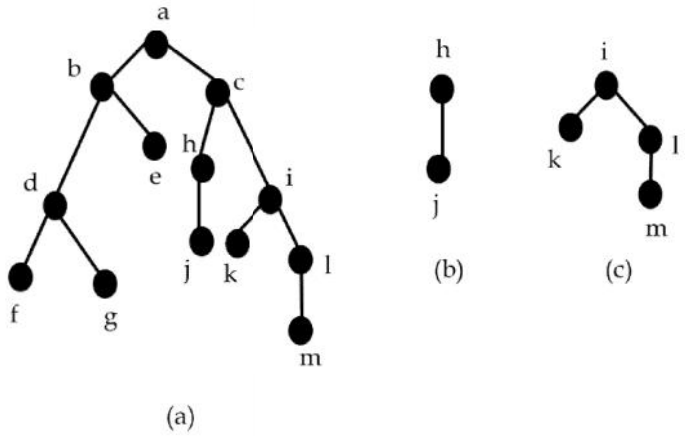


FIGURE 8 A Binary Tree T and Left and Right Subtrees of the Vertex c.

Just as in the case of graphs, there is no standard terminology used to describe trees, rooted trees, ordered rooted trees, and binary trees. This nonstandard terminology occurs because trees are used extensively throughout computer science, which is a relatively young field. The reader should carefully check meanings given to terms dealing with trees whenever they occur.

Trees as Models

Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, and psychology. We will describe a variety of such models based on trees.

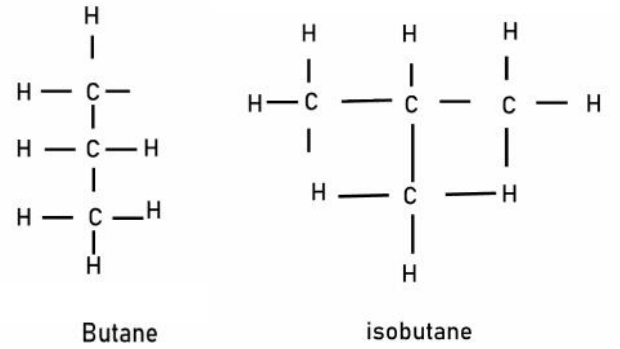


FIGURE 9 The Two Isomers of Butane.

SaturatedHydrocarbonsandTrees Graphscanbeusedtorepresentmolecules, where atoms arerepresentedbyverticesandbondsbetweenthembyedges.
TheEnglishmathematicianArthur Cayley discovered trees in 1857 when he was trying to enumerate the isomers of compounds of the form C_nH_2n+2, which are called saturated hydrocarbons. In graph models of saturated hydrocarbons, each carbon atom is represented by a vertex of degree 4, and each hydrogen atom is represented by a vertex of degree 1. There are 3n+2 verticesinagraphrepresentingacompoundoftheformCnH2n+2.Thenumberofedgesinsucha graph is half the sum of the degrees of the vertices. Hence, there are (4n+2n+2)/2 = 3n+1 edge in this graph. Because the graph is connected and the number of edges is one less than the number of vertices, it must be a tree. The non-isomorphic trees with n vertices of degree 4 and 2n+2 of degree 1 represent the different isomers of CnH2n+2. For instance, when n = 4, there are exactly two non-isomorphic trees of this type (the reader should verify this). Hence, there are exactly two different isomers of C_4H_{10}. Their structures are displayed in Figure 9. These two isomers are called butane and isobutane. ▲

Representing Organizations The structure of a large organization can be modeled using a rooted tree. Each vertex in this tree represents a position in the organization. An edge from one vertex to another indicates that the person represented by the initial vertex is the (direct) boss of the person represented by the terminal vertex. The graph shown in Figure 10 displays such a tree. In the organization represented by this tree, the Director of Hardware Development works directly for the Vice President of R&D. ▲

Computer File Systems Files in computer memory can be organized into directories. A directory can contain both files and subdirectories. The root directory contains the entire filesystem. Thus, a file system may be represented by a rooted tree, where the root represents the root directory, internal vertices represent subdirectories, and leaves represent ordinary files or empty directories. One such file system is shown in Figure 11. In this system, the file khr is in the directory rje. (Note that links to files where the same file may have more than one pathname can lead to circuits in computer file systems.)

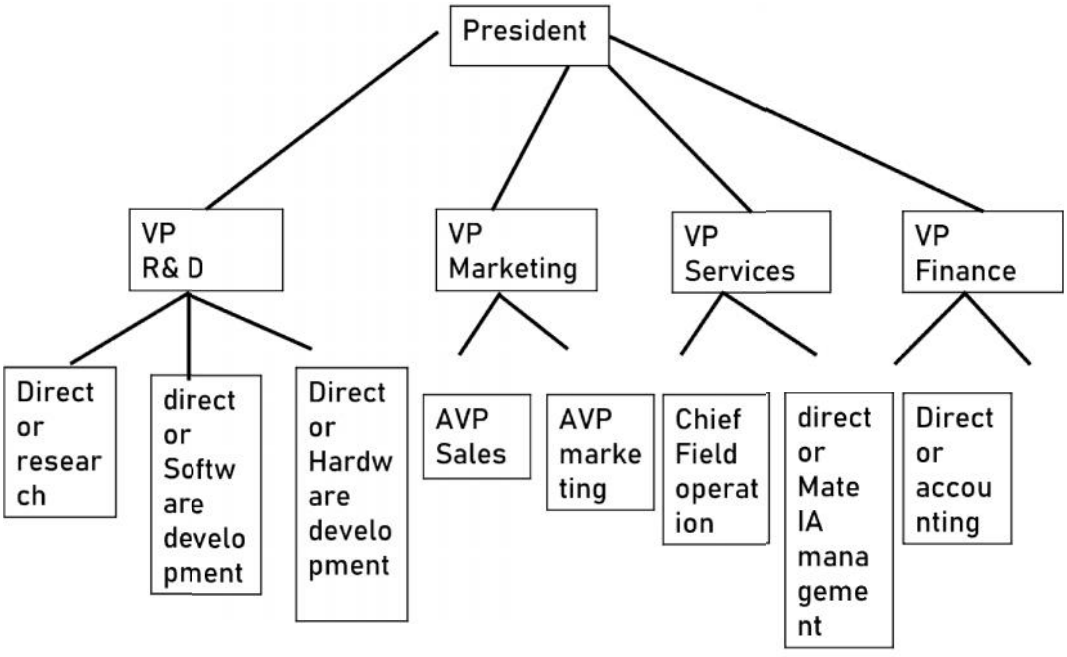


FIGURE 10 An Organizational Tree for a Computer Company.

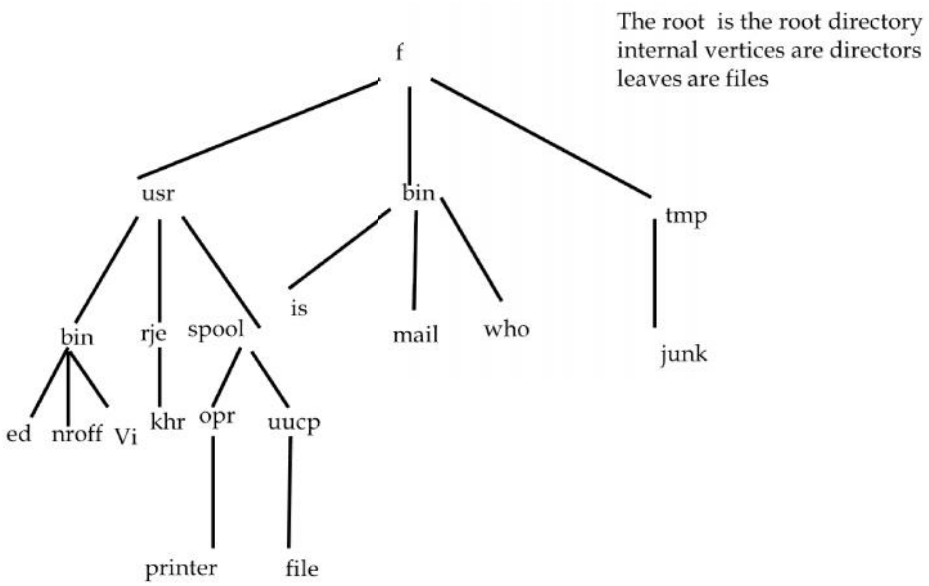


FIGURE 11 A Computer File System.

-connected network is another important way to interconnect processors. The graph ting such a network is a complete binary tree, that is, a full binary tree where every root is ne level. Such a network interconnects n = 2k −1 processor, where k is a positive integer. A r represented by the vertex v that is not a root or a leaf has three two-way connections— e processor represented by the parent of v and two to the processors represented by the dren of v. The processor represented by the root has two two-way connections to the rs represented by its two children. A processor represented by a leaf v has a single two- nection to the parent of v. We display a tree-connected network with seven processors in 2. P_1 P_4 P_5 P_6 P_7 P_2 P_3 We now illustrates how a tree-connected network can be used for computation. In particular, we show how the processors in Figure 12 can be used to add nbers, using three steps. In the first step, we add x_1 and x_2 using P_4, x_3 and x_4 using P_5, x_5 sing P_6,Andx_7 andx_8 usingP_7.Inthesecondstep, weaddx_1 +x_2 andx_3 +x_4 usingP_2 andx_5 +x_6 $_8$ using P_3. Finally, in the third step, we add x_1 +x_2 +x_3 +x_4 and x_5 +x_6 +x_7 +x_8 using P_1. The ps used to add eight numbers compares favorably to the seven steps required to add eight . serially, where the steps are the addition of one number to the sum of the previous in the list ▲

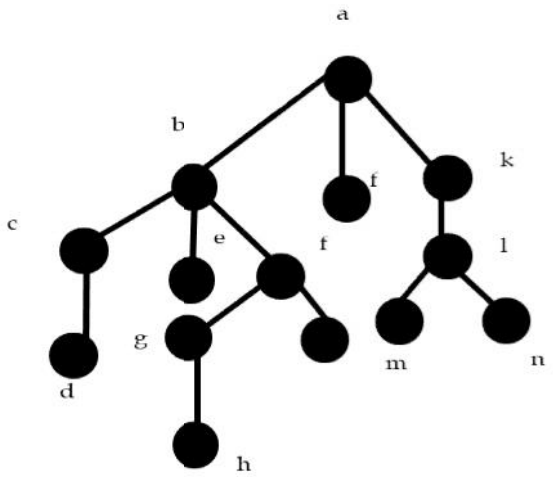


FIGURE 12 A Tree-Connected Network of Seven Processors.

12.3 Properties of Trees

We will often need results relating the numbers of edges and vertices of various types in trees.

THEOREM 2 A tree with n vertices has n−1 edges.

Proof: We will use mathematical induction to prove this theorem. Note that for all the trees here we can choose a root and consider the tree rooted. BASIS STEP: When n = 1, a tree with n = 1 vertex has no edges. It follows that the theorem is true for n = 1. INDUCTIVE STEP: The inductive hypothesis states that every tree with k vertices has k−1 edges, where k is a positive integer. Suppose that a tree T has k+1 vertices and that v is a leaf of T (which must exist because the tree is finite), and let w be the parent of v. Removing from T the vertex v and the edge connecting w to v produces a tree T with k vertices, because the resulting graph is still connected and has no simple circuits. By the inductive hypothesis, T has k−1 edges. It follows that T has k edges because it has one more edge than T, the edge connecting v and w. This completes the inductive step.

Recall that a tree is a connected undirected graph with no simple circuits. So, when G is an undirected graph with n vertices, Theorem 2 tells us that the two conditions (i) G is connected and (ii) G has no simple circuits, imply (iii) G has n−1 edges. Also, when (i) and (iii) hold, then (ii) must also hold, and when (ii) and (iii) hold, (i) must also hold.That is, if G is connected and G has n−1 edges, then G has no simple circuits, so that G is a tree, and if G has no simple circuits and G has n−1 edges, then G is connected, and so is a tree. Consequently, when two of (i), (ii), and (iii) hold, the third condition must also hold and G must be a tree.

COUNTINGVERTICES IN FULL m-ARYTREES the number of vertices in a full m-ary tree with a specified number of internal vertices is determined, as Theorem 3 shows. As in Theorem 2, we will use n to denote the number of vertices in a tree.

THEOREM 3 A full m-ary tree with i internal vertices contains n = mi +1 vertices.

Proof: Every vertex, except the root, is the child of an internal vertex. Because each of the i internal vertices has m children, there are mi vertices in the tree other than the root. Therefore, the tree contains n = mi+1 vertex. Suppose that T is a full m-ary tree. Let i be the number of internal vertices and l the number of leaves in this tree. Once one of n, i, and l is known, the other two quantities are determined. Theorem 4 explains how to find the other two quantities from the one that is known.

THEOREM 4 A full m-ary tree with (i) n vertices has i = (n−1)/m internal vertices and l =[(m−1)n+1]/m leaves, (ii) i internal vertices has n = mi+1 vertices and l = (m−1)i +1 leaves, (iii) l leaves has n = (ml−1)/(m−1) vertices and i = (l−1)/(m−1) internal vertices.

Proof: Let arepresentthenumberofvertices, i thenumberofinternalvertices, andl thenumber of leaves. The three parts of the theorem can all be proved using the equality given inTheorem 3, that is, n = mi+1, together with the equality n = l+i, which is true because each vertex is either a leaf or an internal vertex. We will prove part (i) here. The proofs of parts (ii) and (iii) are left as exercises for the reader. Solving for i in n = mi+1givesi = (n−1)/m. Then inserting this expression for i into the equation n = l+i shows that l = n−i = n−(n−1)/m =[(m−1) n+1]/m.

Example 9 illustrates how Theorem 4 can be used.

Suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to four other people. Some people do this, but others do not send any letters. How many people have seen the letter, including the first person, if no one receives more than one letter and if the chain letter ends after there have been 100 people who read it but did not send it out? How many people sent out the letter?

Solution: The chain letter can be represented using a 4-ary tree. The internal vertices correspond to people who sent out the letter, and the leaves correspond to people who did not send it out. Because 100 people did not send out the letter, the number of leaves in this rooted tree is l = 100. Hence, part(iii)ofTheorem4showsthatthenumberofpeoplewhohaveseentheletter is n = (4·100−1)/(4−1) = 133. Also, the number of internal vertices is 133−100 = 33, so 33 people sent out the letter.

Solution: The chain letter can be represented using a 4-ary tree. The internal vertices correspond to people who sent out the letter, and the leaves correspond to people who did not send it out. Because 100 people did not send out the letter, the number of leaves in this rooted tree is l = 100. Hence, part(iii)ofTheorem4showsthatthenumberofpeoplewhohaveseentheletter is n = (4·100−1)/(4−1) = 133. Also, the number of internal vertices is 133−100 = 33, so 33 people sent out the letter. ▲

BALANCED m-ARYTREES It is often desirable to use rooted trees that are "balanced" so thatthesubtreesateachvertexcontainpathsofapproximatelythesamelength. Somedefinitions will make this concept clear. The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex. The level of the root is defined to be zero. The height of a rooted tree is the maximum of the levels of vertices. In other words, the height of a rooted tree is the length of the longest path from the root to any vertex.

Find the level of each vertex in the rooted tree shown in Figure 13. What is the height of this tree?

Solution: The root a is at level 0. Vertices b, j, and k are at level 1. Vertices c, e, f, and l are at level 2. Vertices d, g, i, m, and n are at level 3. Finally, vertex h is at level 4. Because the largest level of any vertex is 4, this tree has height 4. ▲ A rooted m-ary tree of height h is balanced if all leaves are at levels h or h−1.

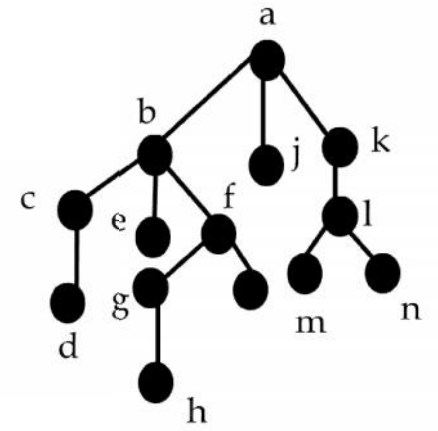


FIGURE 13 A Rooted Tree.

Which of the rooted trees shown in Figure 14 are balanced?

Solution: T_1 is balanced, because all its leaves are at levels 3 and 4. However, T_2 is not balanced, because it has leaves at levels 2, 3, and 4. Finally, T_3 is balanced, because all its leaves are at level 3.

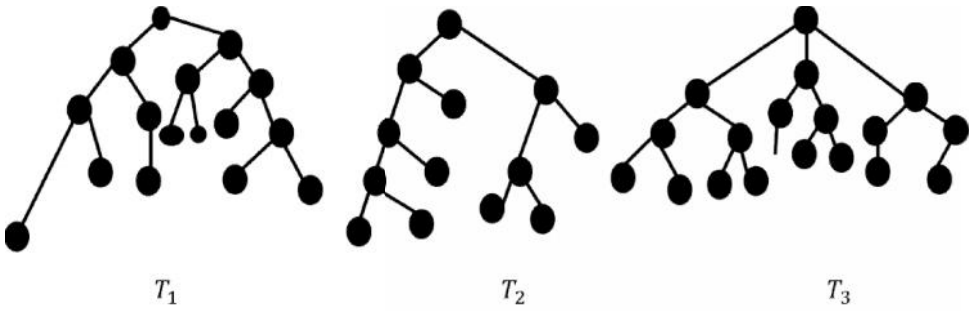


FIGURE 14 Some Rooted Trees.

ABOUNDFORTHENUMBEROFLEAVESINANm –ARYTREE It is often useful to have an upper bound for the number of leaves in an m-ary tree. Theorem 5 provides such a bound in terms of the height of the m-ary tree.

THEOREM 5 There are at most m^hleaves in anm-ary tree of height h.

Proof: The proof uses mathematical induction on the height. First, consider m-ary trees of height 1. These trees consist of a root with no more than m children, each of which is a leaf. Hence, there are

no more than $m^1 = m$ leaves in anm-ary tree of height 1. This is the basis step of the inductive argument.

Nowassumethattheresultistrueforallm–arytreesofheightlessthanh; thisistheinductive hypothesis. Let T be an m-ary tree of height h. The leaves of T are the leaves of the subtrees of T obtained by deleting the edges from the root to each of the vertices at level 1, as shown in Figure 15. Each of these subtrees has height less than or equal to h−1. So, by the inductive hypothesis, each of these rooted trees has at most m^{h-1} leaf. Because there are at most m such subtrees, each with a maximum of m^{h-1} leaf, there are at most $m \cdot m^{h-1} = m^h$ leaves in the rooted tree,this finishes the inductive argument.

COROLLARY 1 If an m-ary tree of height h has l leaves, then $h \geq [\log_m l]$. If the m-ary tree is full and balanced, then $h = [\log_m l]$. (We are using the ceiling function here. Recall that [x] is the smallest integer greater than or equal to x.)

Proof: We know that $l \leq m^h$ from Theorem 5. Taking logarithms to the base m shows that $\log_m l \leq h$. Becausehisan integer, wehaveh $\geq [\log_m l]$. Now suppose thatthetreeisbalanced.

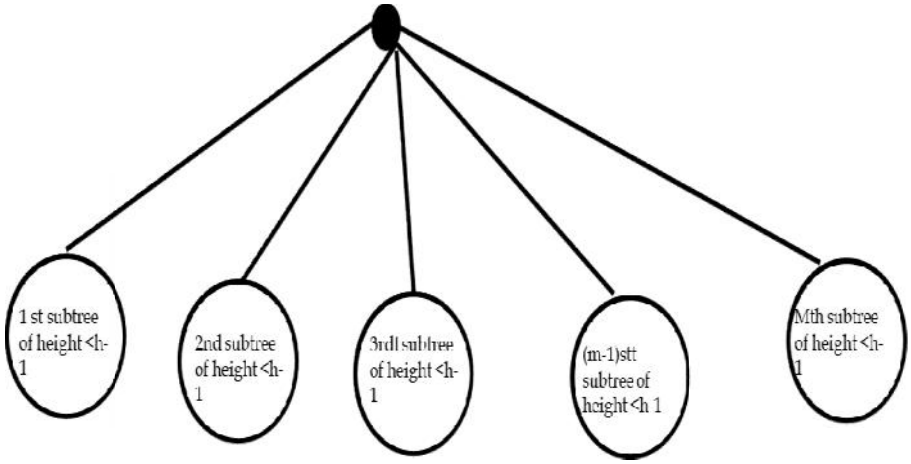
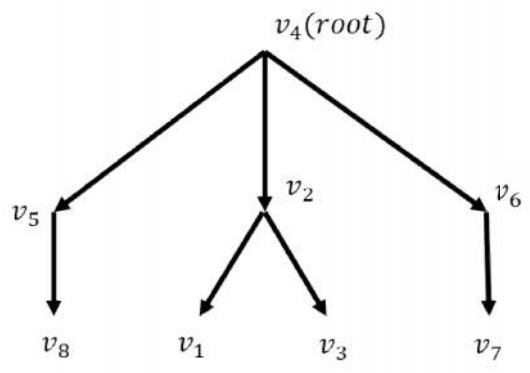


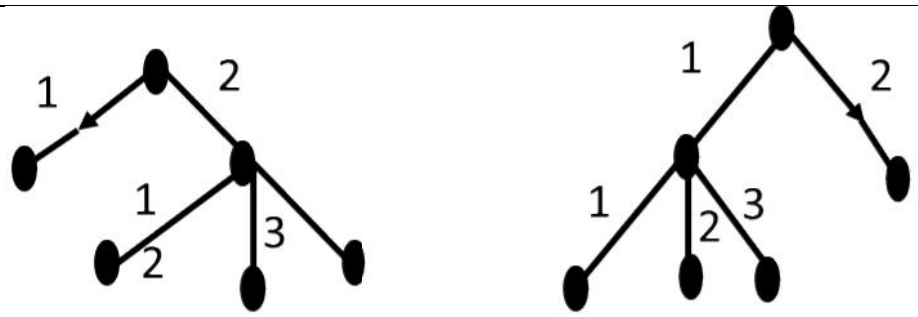
FIGURE 15 The Inductive Step of the Proof.

Theneachleafisatlevelhorh−1, andbecausetheheightish, thereisatleast oneleafatlevelh. It follows that there must be more than m^{h-1}leaf. Because $l \leq m^h$, we have $m^h-1 < l \leq m^h$. Taking logarithms to the base m in this inequality gives $h-1 < \log_m l \leq h$. Hence, $h = [\log_m l]$.

Definition: Two ordered trees are said to be isomorphic if (i) there exists a one-to-one correspondence between their vertices and edges and that preserves the incident relation (ii) labels

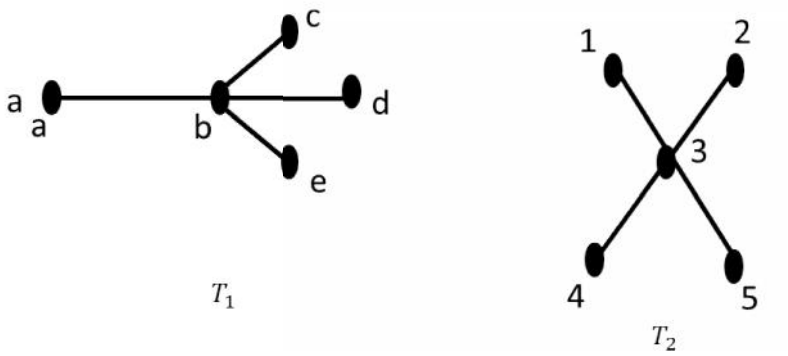


of the corresponding edges match, In view of this definition, the ordered trees



are not isomorphic.

Show that the tree T_1 and T_2 shown in the diagram below are isomorphic.



Solution: We observe that in the tree T_1,

$$\deg(b) = 4$$

In the tree T_2,

$$\deg(3) = 4$$

Further deg(a) = deg (1) = 1, deg c = deg (2), deg(d) = deg(4) = deg(e) = 1 = deg(5). Thus, we may define a function f from the vertices of T_1 to the vertices of T_2 by

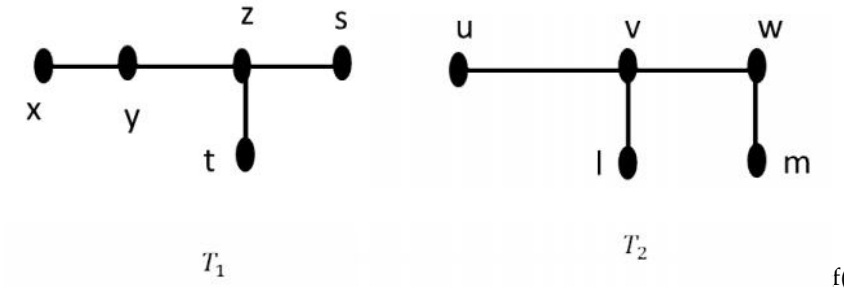
$$f(a) = 1, f(b) = 3, f(c) = 2, f(d) = 4, f(e) = 5$$

This is a one-to-one and onto function. Also, adjacency relation is preserved because if v_i and v_j are adjacent vertices in T_1, then $f(v_i)$ and $f(v_j)$ are adjacent vertices in T_2. Hence T_1 is isomorphic to T_2.

Show that the tree T_1 and T_2, shown in the figure below are isomorphic

Solution: Let f be a function defined by

$$f(z) = v$$



$$f(b) = w$$

$$f(x) = m$$

$$f(s) = u$$

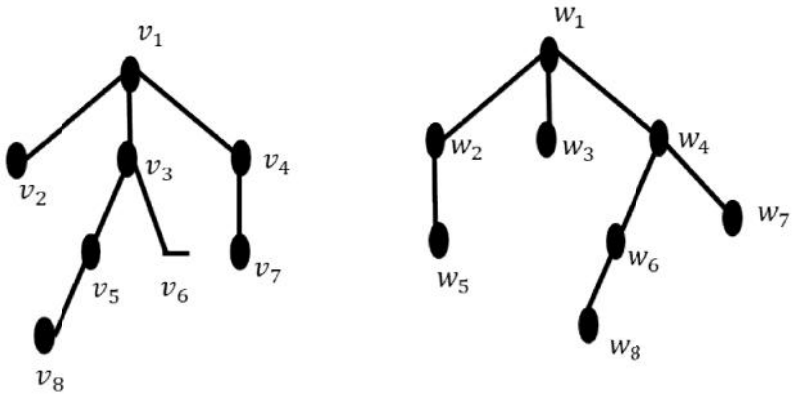
$$f(t) = l.$$

Then f is a one to one onto mapping which preserves adjacency. HenceT_1 and T_2 are isomorphic. Definition: Let T_1 and T_2 be rooted tree with roots r_1 and r_2 respectively. Then T_1 and T_2 are isomorphic if there exists a one-to-one, onto function f from the vertex set of T_1 to the vertex set of T_2 such that

Vertices v_i and v_j are adjacent in T_1 if and only if the vertices $f(v_i)$ and $f(v_j)$ are adjacent in T_2.

(ii) $f(r_1) = r_2$ the function is then called an isomorphism.

Show that the tree T_1 and T_2 are isomorphic.



Solution: We observe that T_1 and T_2 are rooted tree.

Define f: (Vertex set of T_1) → (Vertex set of T_2) by

$f(v_1) = w_1$

$f(v_2) = w_3$

$f(v_3) = w_4$

$f(v_4) = w_2$

$f(v_5) = w_6$

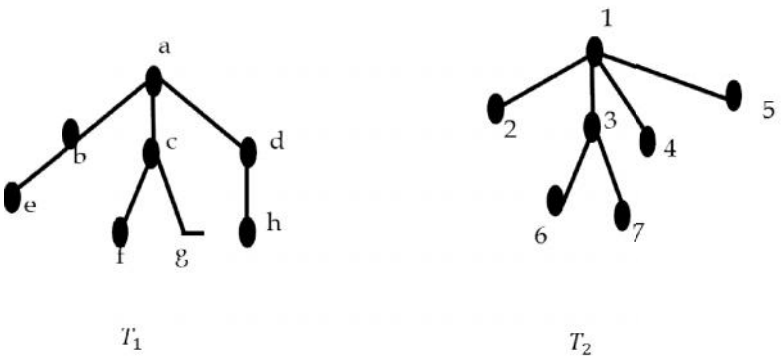
$f(v_6) = w_7$

$f(v_7) = w_5$

$f(v_8) = w_8$

Thenf is one-to- one and adjacency relation is preserved. Hence f is an isomorphism and so the rooted tree T_1 and T_2 are isomorphic

Show that the rooted tree shown below are not isomorphic:



Solution: We observe that the degree of root in T_1 is 3, whereas the degree of root in T_2 is 4. Hence T_1 is not isomorphic to T_2.

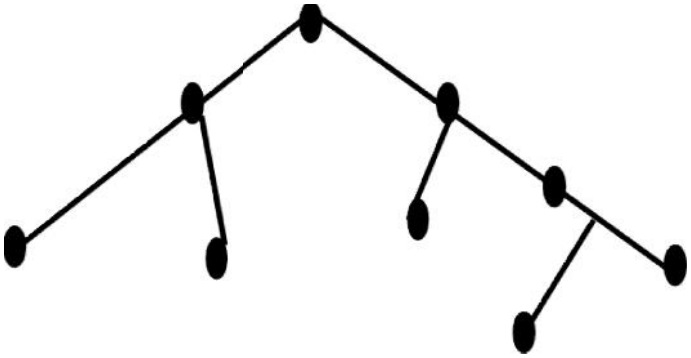
Definition: An ordered tree in which every branch node has almost n offspring's is called a n-ary tree (or n-tree).

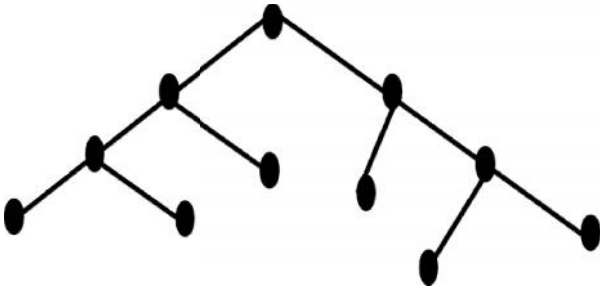
Definition: An n-ary tree is said to be fully n-ary tree (complete n-ary tree or regular n ary tree) if every branch node has exactly n offspring.

Definition: An ordered tree in which every branch node has almost 2 offspring's is called a binary tree (or 2 - tree).

Definition: A binary tree in which every branch node (internal vertex) has exactly two offspring's is called a fully binary tree.

For example, the tree given below is a binary tree, whereas the tree shown below is a fully binary tree.





Definition: Let T_1 and T_2 be binary trees roots r_1 and r_2 respectively. Then T_1 and T_2 are isomorphic if there is a one to one, onto function f from the vertex set of T_1 to the vertex set of T_2 satisfying

Vertices v_i and v_j are adjacent in T_1 if and only if the vertices $f(v_i)$ and $f(v_j)$ are adjacent in T_2.

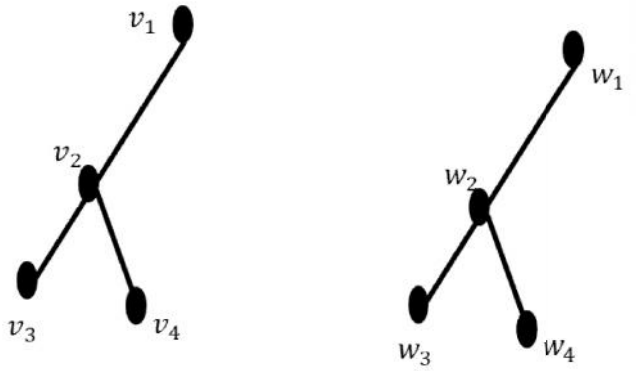
(ii) $f(r_1) = r_2$

(iii) v is a left child of w in T_1 if and only if f(v) is a left child of f(w) in T_2

(iv) v is a right child of w in T_1 if and only if f(v) is a right child of f(w) in T_2.

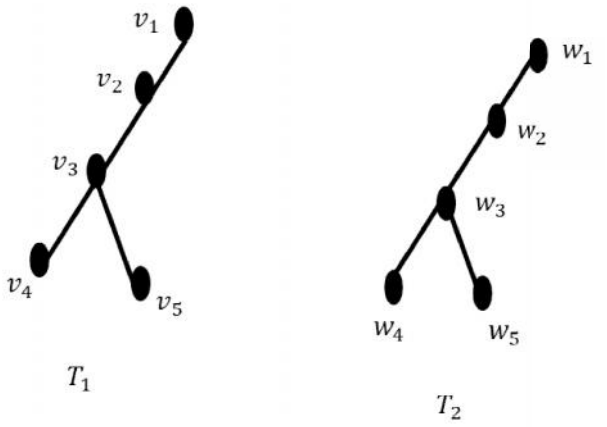
The function f is then called an isomorphism between binary tree T1 and T2

Show that the trees given below are isomorphic.



Solution: Define f by $f(v_i) = w_i$, i = 1, 2, 3, 4. Then f satisfies all the properties for isomorphism. Hence T_1 and T_2 are isomorphic.

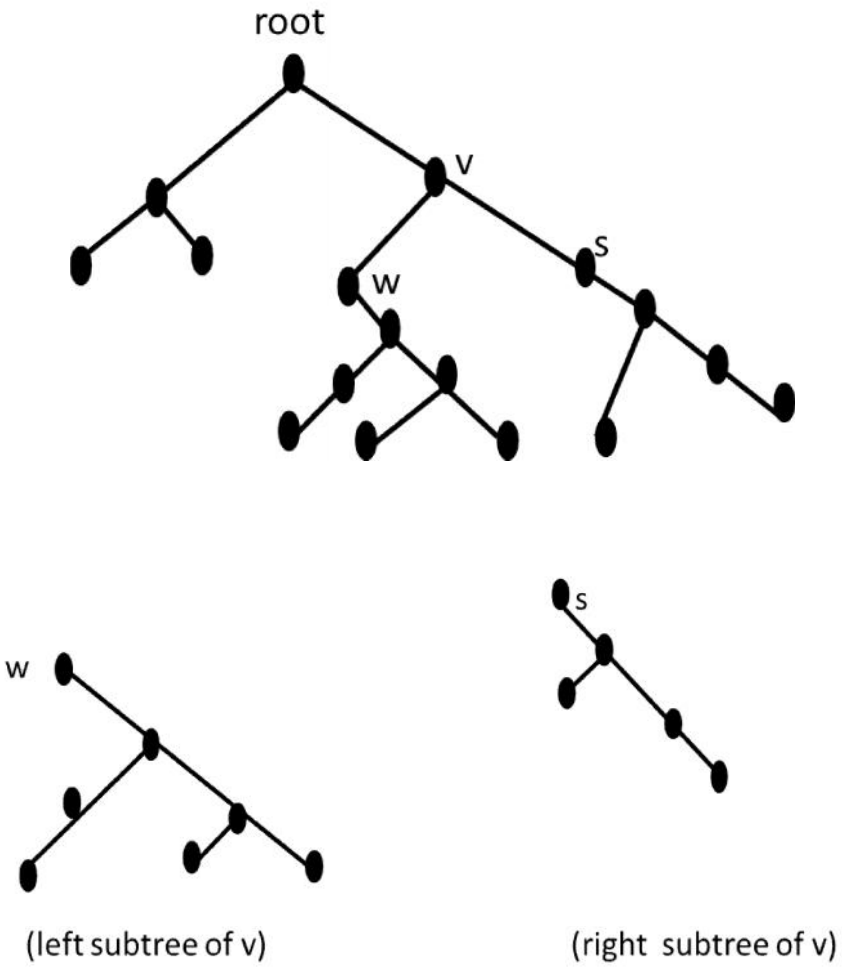
Show that the trees given below are not isomorphic.



Solution: Since the root v1 in T1 has a left child but the root w1 in T2 has no left child, the binary trees are not isomorphic.

Definition: Let v be a branch node of a binary tree T. The left subtree of v is the binary tree whose root is the left child of v, whose vertices consists of the left child of v and all its descendents and whose edges consists of all those edges of T that connects the vertices of the left subtree together. The right subtree can be defined analogously.

For example, the left subtree and the right subtree of v in the tree (shown below):

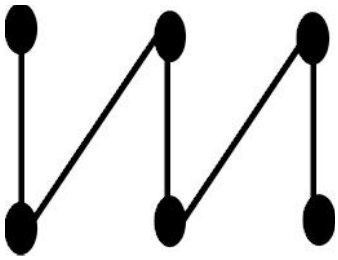


Summary

- We have learned what are Trees
- We have learned what different parts of trees

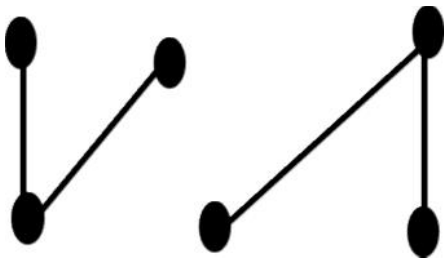
Self Assessment

1. For the graph below which of the following statement is true?



A. This graph is connected and has no simple circuits, so it is a tree.
B. This graph Is not connected and has no simple circuits, so it is a tree.
C. This graph is undirected graph.
D. This graph is connected acyclic undirected graph.

2. For the graph below which of the following statement is true?



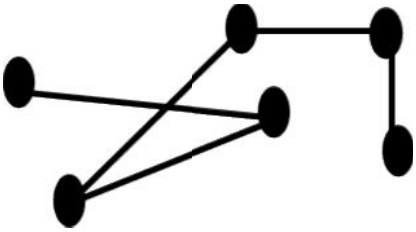
A. This graph is connected, so it is not a tree.

B. This graph is undirected graph

C. This graph is directed graph

D. This graph is not connected, so it is not a tree.

3. For the graph below which of the following statement is true?



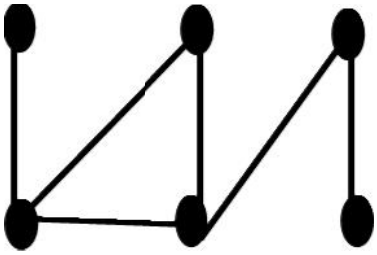
A. This graph is connected and has no simple circuits, so it is a tree.

B. This graph is connected, so it is not a tree.

C. This graph is directed graph

D. This graph is not connected, so it is not a tree.

4. For the graph below which of the following statement is true?



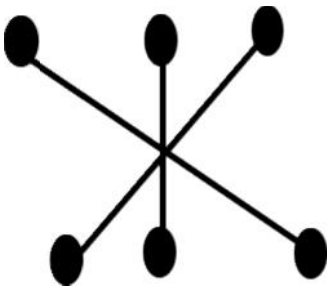
A. This graph has a circuit, so it is a tree.

B. This graph is not connected, so it is not a tree.

C. This graph has a simple circuit, so it is not a tree.

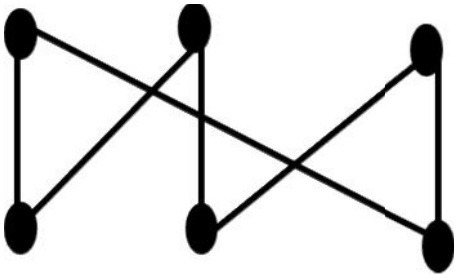
D. This graph is directed graph

5. For the graph below which of the following statement is true?



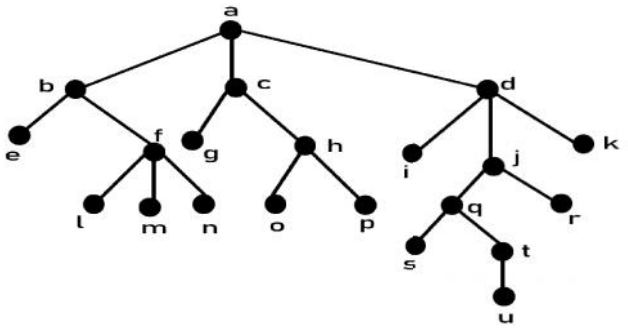
A. This graph is connected so it is a tree.
B. This graph is not connected and has no simple circuits, and it is not a tree.
C. This graph has a circuit, so it is a tree.
D. This graph is connected, so it is not a tree.

6. For the graph below which of the following statement is true?



A. This graph has a simple circuit, so it is not a tree.
B. This graph is connected, so it is not a tree.
C. This graph does not have a simple circuit, so it is not a tree.
D. This graph is not connected and have a circuit

7. Which vertex is the root in the rooted tree illustrated below?



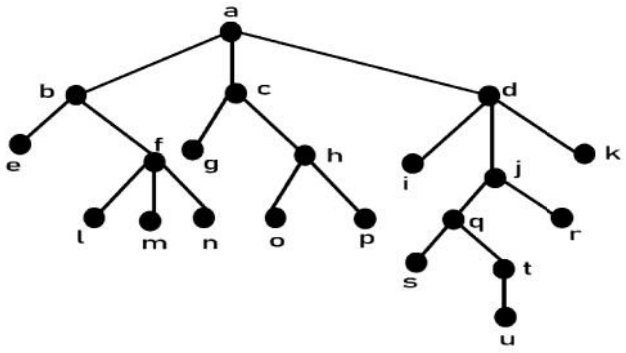
A. Vertex b is the root, since it is drawn at the top.

B. Vertex a is the root, since it is drawn at the top.

C. Vertex c is the root, since it is drawn at the top.

D. Vertex u is the root, since it is drawn at thebottom.

8. Which vertices are internal in the rooted tree illustrated below?



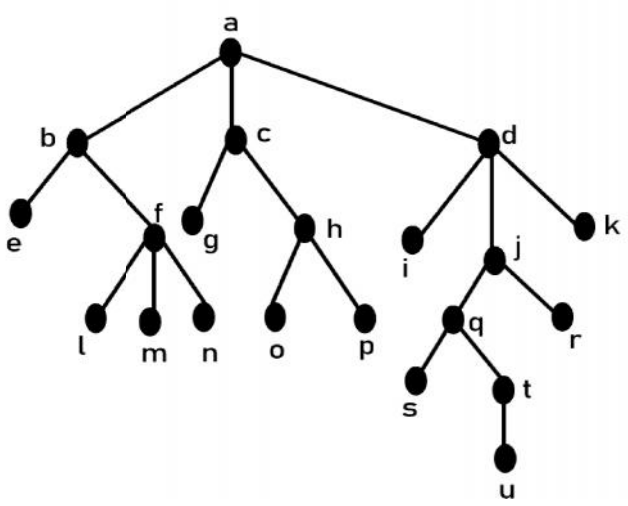
A. The internal vertices are the vertices with children, namely a, b, c, d, f, e, g, I, m and t.

B. The internal vertices are the vertices with children, namely a, b, c, d, f, k, l, j, q, and t.

C. The internal vertices are the vertices with children, namely a, b, c, d, f, h, j, q, and t.

D. The internal vertices are the vertices with children, namely a, b, c, d.

9. Which vertices are leaves in the rooted tree illustrated below?



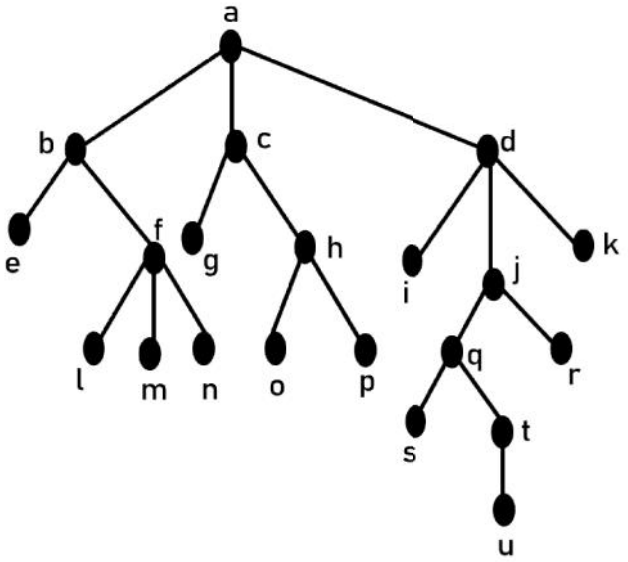
A. The leaves are the vertices without children, namely e, g, i, k, l, m, n, o, p, r, s, and u.

B. The leaves are the vertices with children, namely e, g, i, k, l, m, n, o, p, r, s, and j.

C. The leaves are the vertices without children, namely e, g, i, k, l, m, n, o, p, r, s, and l.

D. The leaves are the vertices with children, namely e, g, i, k, l, m, p, o, p, r, s, and m.

10. Which vertices are children of j in the rooted tree illustrated below?



A. The children of j are the vertices adjacent to k and below m, namely q and p.

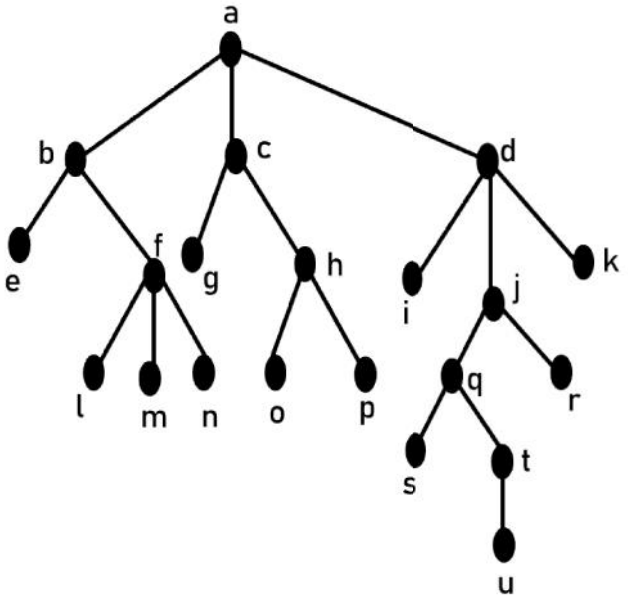
B. The children of j are the vertices adjacent to k and below m, namely q and p.

C. The children of c are the vertices adjacent to a and below b, namely q and m.

D. The children of h are the vertices adjacent to e and below k, namely q and l.

E. The children of j are the vertices adjacent to j and below j, namely q and r.

11. Which vertex is the parent of h in the rooted tree illustrated below?



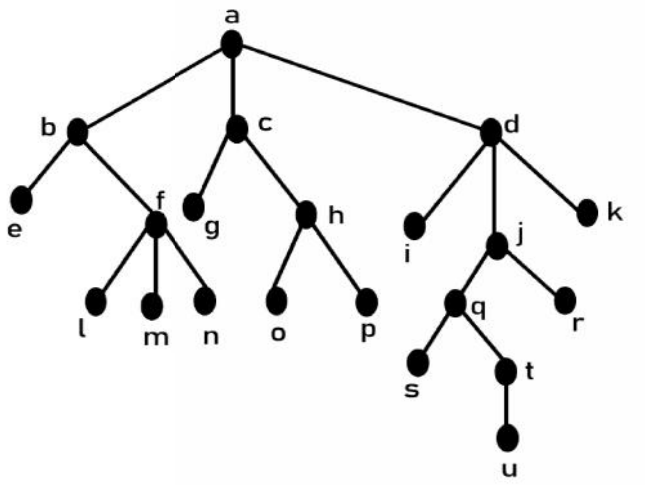
A. The parent of k is the vertex adjacent to h and above m, namely c.

B. The parent of his the vertex adjacent to h and above d, namely l.

C. The parent of h is the vertex adjacent to h and above h, namely c.

D. The parent of b is the vertex adjacent to a and above c, namely e.

12. Which vertices are siblings of O in the rooted tree illustrated below?



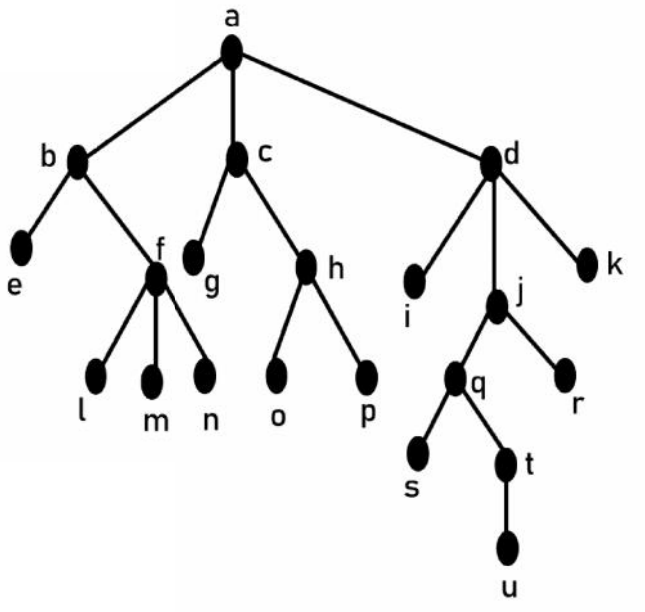
A. Vertex o has only two siblings, namely o, which is the other child of o's parent, k.

B. Vertex o has only three siblings, namely s, which is the other child of o's parent, k.

C. Vertex o has only four siblings, namely a, which is the other child of o's parent, l.

D. Vertex o has only one sibling, namely p, which is the other child of o's parent, h.

13. Which vertices are ancestors of m in the rooted tree illustrated below?



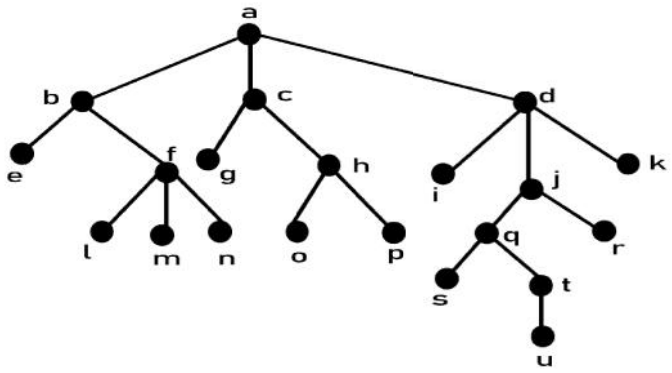
A. The ancestors of m are all the vertices on the unique simple path from m back to the root, namely l, h, and q.

B. The ancestors of m are all the vertices on the unique simple path from m back to the root, namely f, b, and l.

C. The ancestors of m are all the vertices on the unique simple path from m back to the root, namely f, b, and a.

D. The ancestors of m are all the vertices on the unique simple path from m back to the root, namely s, s, andl.

14. Which vertices are descendants of b in the rooted tree illustrated below?



A. The descendants of c are all the vertices that have b as an ancestor, namely d, a, k, l, and q.

B. The descendants of b are all the vertices that have b as an ancestor, namely e, f, l, m, and n.

C. The descendants of g are all the vertices that have b as an ancestor, namely k, l, b, m, and k.

D. The descendants of c are all the vertices that have c as an ancestor, namely n, m, o, p, and q.

15. A tree with n vertices has

A. n−1 edges

B. n−2 edges

C. n−3 edges

D. n−4 edges

Answer for Self Assessment

1.	A	2.	D	3.	A	4.	C	5.	B
6.	C	7.	B	8.	C	9.	A	10.	D
11.	C	12.	D	13.	C	14.	B	15.	A

Review Question

1. Show that a simple graph is a tree if and only if it is connected but the deletion of any of its edges produces a graph that is not connected.
2. A full m-ary tree T has 81 leaves and height 4. Give the upper and lower bounds for m.
3. A full m-ary tree T has 81 leaves and height 4. What is m if $T$is also balanced?
4. Show that a full m-ary tree with I internal vertices contains $n = mi + 1$ vertices.
5. How many non-isomorphic unrooted trees are there with four vertices?

Further Reading

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 13: Spanning Trees

CONTENTS

Objectives

13.1 Tree graphs

13.2 Spanning Trees

13.3 Minimum Spanning Trees

13.4 Introduction

13.5 Depth-First Search

13.6 Breadth-First Search

13.7 Depth-First Search in Directed Graphs

13.8 Algorithms for Minimum Spanning Trees

Summary

Self Assessment

Answer for Self Assessment

Review Question

Further Reading

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- understand what Spanning Trees
- understand what is Kruskal's Algorithm.
- understand what is Prim's Algorithm.

13.1 Tree graphs

A graph T is called a tree if T is connected and T has no cycles. A forest G is a graph with no cycles; hence the connected components of a forest G are trees. A graph without cycles is said to be cycle-free. The tree consisting of a single vertex with no edges is called the degenerate tree.Consider a tree T. Clearly, there is only one simple path between two vertices of T; otherwise, the two paths would form a cycle. Also:(a) Suppose there is no edge {u, v} in T and we add the edge e = {u, v} to T. Then the simple path from u to v in T and e will form a cycle; hence T is no longer a tree.(b) On the other hand, suppose there is an edge e = {u, v} in T, and we delete e from T . Then T is no longer connected (since there cannot be a path from u to v); hence T is no longer a tree.

The following theorem applies when our graphs are finite.

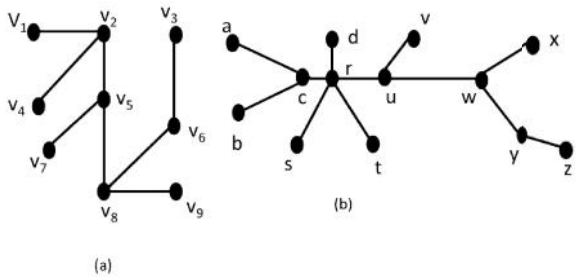
Theorem: Let G be a graph with n > 1 vertices. Then the following are equivalent:

(i) G is a tree.

(ii) G is a cycle-free and has n − 1 edges.

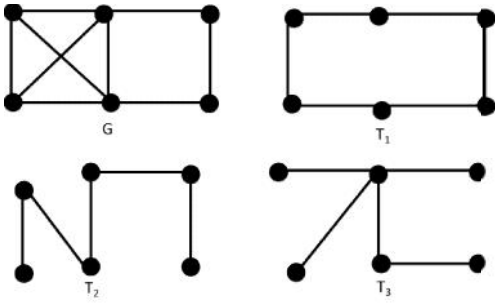
(iii) G is connected and has n − 1 edges.

This theorem also tells us that a finite tree T with n vertices must have n − 1 edges. For example, the tree in Fig.(a) has 9 vertices and 8 edges, and the tree in Fig.(b) has 13 vertices and 12 edges.



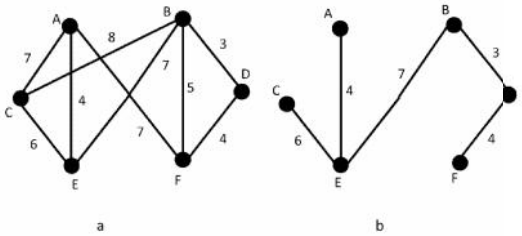
13.2 Spanning Trees

A subgraph T of a connected graph G is called a spanning tree of G if T is a tree and T includes all the vertices of G. Figure below shows a connected graph G and spanning trees T1, T2, and T3 of G.



13.3 Minimum Spanning Trees

Suppose G is a connected weighted graph. That is, each edge of G is assigned a nonnegative number called the weight of the edge. Then any spanning tree T of G is assigned a total weight obtained by adding the weights of the edges in T . A minimal spanning tree of G is a spanning tree whose total weight is as small as possible.



Algorithms 1 and 2, which appear in Fig. below, enable us to find a minimal spanning tree T of a connected

Algorithm 1: the input is a connected weighted graph G with n vertices.

Step1. Arrange the edges of G in the order of decreasing weights

Step2. Proceeding sequentially deletes each edge that does not disconnect the graph until n-1 edges remain.

Step 3. Exit

Algorithm 2 (Kruskal): the input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in the order of increasing weights

Step 2. Starting only with the vertices of G and proceeding sequentially, add each edge with doe not result in a cycle until n-1 edges are added

Step 3. Exit

The weight of a minimal spanning tree is unique, but the minimal spanning tree itself is not. Different minimal spanning trees can occur when two or more edges have the same weight. In such

a case, the arrangement of the edges in Step 1 of Algorithms 1 or 2 is not unique and hence may result in different minimal spanning trees as illustrated in the following example.

Find a minimal spanning tree of the weighted graph Q in Fig. (a). Note that Q has six vertices, so a minimal spanning tree will have five edges.

(a) Here we apply Algorithm 1

First we order the edges by decreasing weights, and then we successively delete edges without disconnecting Q until five edges remain. This yields the following data:

Edges BC AF AC BE CE BF AE DF BD

Weight 8 7 7 7 6 5 4 4 3

Delete Yes YesYes No No Yes

Thus the minimal spanning tree of Q which is obtained contains the edges

BE, CE, AE, DF , BD

The spanning tree has weight 24 and it is shown in Fig. (b).

(b) Here we apply Algorithm 2.

First we order the edges by increasing weights, and then we successively add edges without forming any cycles until five edges are included. This yields the following data:

Edges BD AE DF BF CE AC AF BE BC

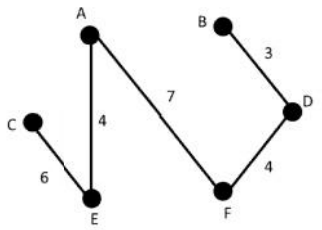
Weight 3 4 4 5 6 7 7 7 8

Add? YesYesYes No Yes No Yes

Thus the minimal spanning tree of Q which is obtained contains the edges

BD, AE, DF , CE, AF

The spanning tree appears in Fig. (c). Observe that this spanning tree is not the same as the one obtained using Algorithm 1 as expected it also has weight 24.



c

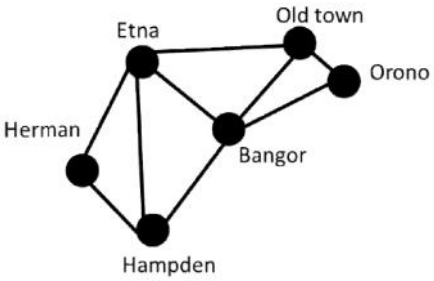
13.4 Introduction

Consider the system of roads in Maine represented by the simple graph shown in Figure 1(a).

The only way the roads can be kept open in the winter is by frequently plowing them. Thehighway department wants to plow the fewest roads so that there will always be cleared roadsconnecting any two towns. How can this be done?

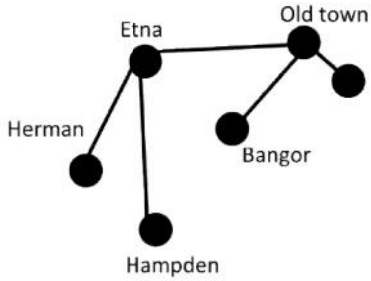
At least five roads must be plowed to ensure that there is a path between any two towns.Figure 1(b) shows one such set of roads. Note that the subgraph representing these roads is atree because it is connected and contains six vertices and five edges.

This problem was solved with a connected subgraph with the minimum number of edgescontaining all vertices of the original simple graph. Such a graph must be a tree.



(a)

Figure 1 (a) A road system and (b) a set of roads to plow



(b)

Figure 2 The simple graph G

DEFINITION 1 Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing very vertex of G.

A simple graph with a spanning tree must be connected, because there is a path in thespanning tree between any two vertices. The converse is also true; that is, every connectedsimple graph has a spanning tree. We will give an example before proving this result.

Find a spanning tree of the simple graph G shown in Figure 2.

Solution: The graph G is connected, but it is not a tree because it contains simple circuits.

Remove the edge {a, e}. This eliminates one simple circuit, and the resulting subgraph is stillconnected and still contains every vertex of G. Next remove the edge {e, f} to eliminate asecond simple circuit. Finally, remove edge {c, g} to produce a simple graph with no simplecircuits. This subgraph is a spanning tree, because it is a tree that contains every vertex of G.

The sequence of edge removals used to produce the spanning tree is illustrated in Figure 3.

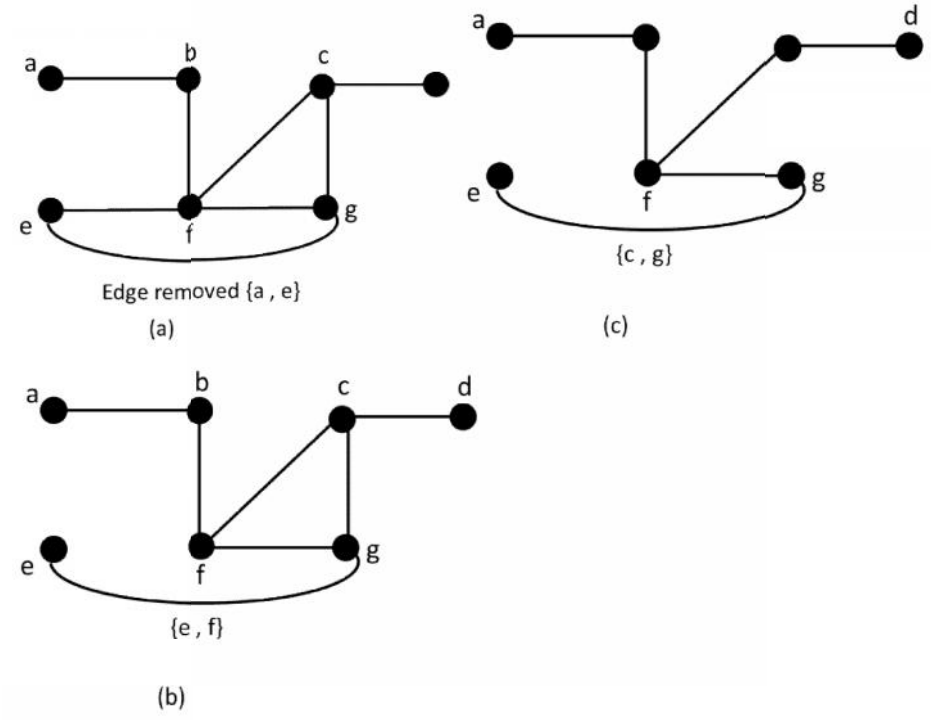


Figure 3 producing a spanning tree for G by removing edges that form simple circuits

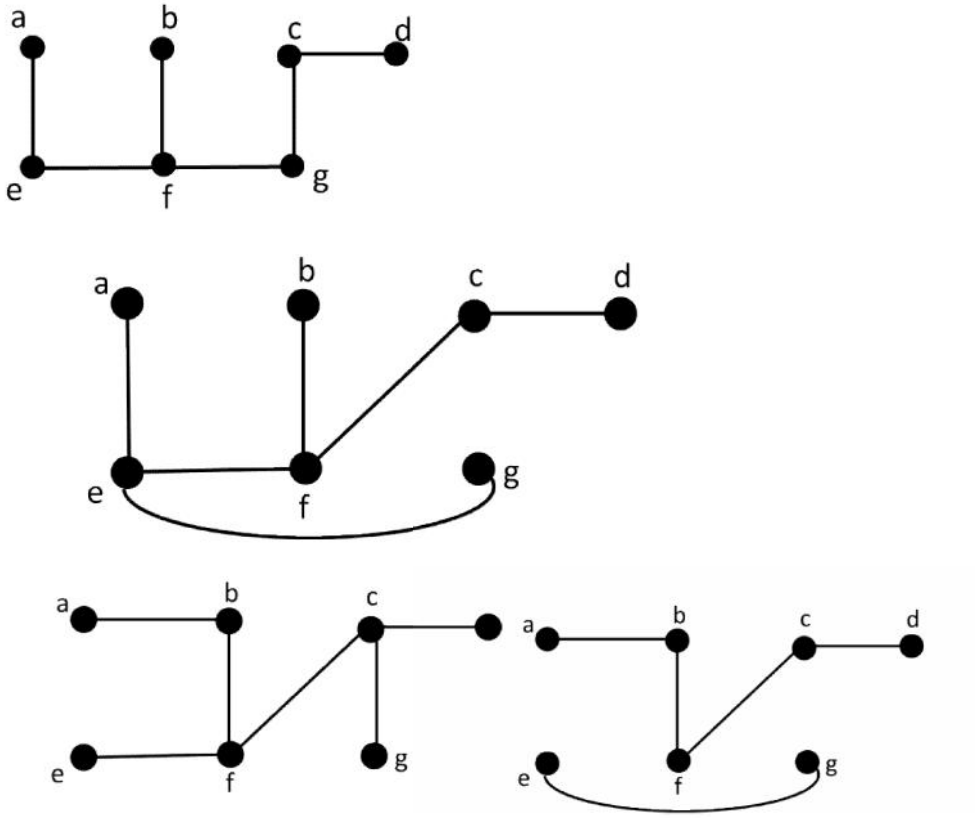


Figure 4 spanning trees of G

The tree shown in Figure 3 is not the only spanning tree of G. For instance, each of the trees shown in Figure 4 is a spanning tree of G. ▲

THEOREM 1 A simple graph is connected if and only if it has a spanning tree.

Proof: First, suppose that a simple graph G has a spanning tree T. T contains every vertex of G. Furthermore, there is a path in T between any two of its vertices. Because T is a subgraph of G, there is a path in G between any two of its vertices. Hence, G is connected.

Now suppose that G is connected. If G is not a tree, it must contain a simple circuit. Removean edge from one of these simple circuits. The resulting subgraph has one fewer edge but stillcontains all the vertices of G and is connected. This subgraph is still connected because whentwo vertices are connected by a path containing the removed edge, they are connected by a pathnot containing this edge. We can construct such a path by inserting into the original path, atthe point where the removed edge once was, the simple circuit with this edge removed. If thissubgraph is not a tree, it has a simple circuit; so as before, remove an edge that is in a simplecircuit. Repeat this process until no simple circuits remain. This is possible because there areonly a finite number of edges in the graph. The process terminates when no simple circuitsremain. A tree is produced because the graph stays connected as edges are removed. This treeis a spanning tree because it contains every vertex of G.

Spanning trees are important in data networking, as Example 2 shows.

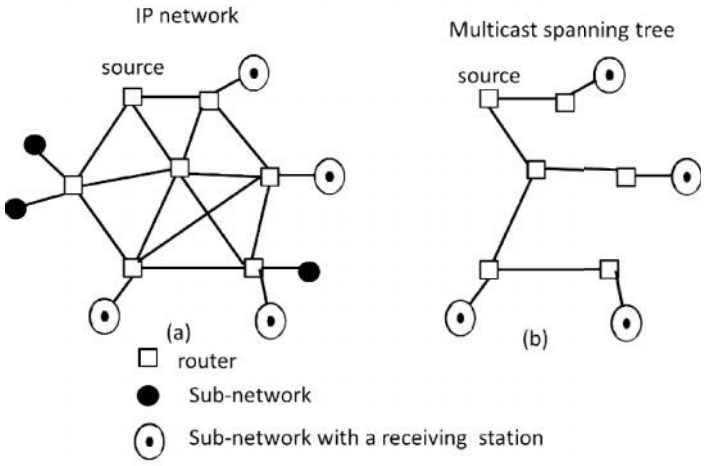
1. IP Multicasting Spanning trees play an important role in multicasting over Internet Protocol(IP) networks. To send data from a source computer to multiple receiving computers, each ofwhich is a subnetwork, data could be sent separately to each computer. This type of networking, Calleduncasing, is inefficient, because many copies of the same data are transmitted over thenetwork. To make the transmission of data to multiple receiving computers more efficient,

IPmulticasting is used. With IP multicasting, a computer sends a single copy of data over thenetwork, and as data reaches intermediate routers, the data are forwarded to one or more otherrouters so that ultimately all receiving computers in their various subnetworks receive these data.

(Routers are computers that are dedicated to forwarding IP datagrams between subnetworks in

a network. In multicasting, routers use Class D addresses, each representing a session thatreceiving computers may join)

For data to reach receiving computers as quickly as possible there should be no loops(which in graph theory terminology are circuits or cycles) in the path that data take through thenetwork. That is, once data have reached a particular router, data should never return to this



Router,To avoid loops, the multicast routers use network algorithms to construct a spanning treein the graph that has the multicast source, the routers, and the subnetworks containing receivingcomputers as vertices, with edges representing the links between computers and/or routers.

The root of this spanning tree is the multicast source. The subnetworks containing receivingcomputers are leaves of the tree. (Note that subnetworks not containing receiving stations are

Toincluded in the graph.) This is illustrated in Figure 5.

13.5 Depth-First Search

The proof of Theorem 1 gives an algorithm for finding spanning trees by removing edges fromsimple circuits. This algorithm is inefficient, because it requires that simple circuits be identified.

Instead of constructing spanning trees by removing edges, spanning trees can be built up bysuccessively adding edges. Two algorithms based on this principle will be presented here.

We can build a spanning tree for a connected simple graph using depth-first search. Wewill form a rooted tree, and the spanning tree will be the underlying undirected graph of thisrooted tree. Arbitrarily choose a vertex of the graph as the root. Form a path starting at thisvertex by successively adding vertices and edges, where each new edge is incident with the lastvertex in the path and a vertex not already in the path. Continue adding vertices and edges to thispath as long as possible. If the path goes through all vertices of the graph, the tree consisting ofthis path is a spanning tree. However, if the path does not go through all vertices, more verticesand edges must be added. Move back to the next to last vertex in the path, and, if possible, forma new path starting at this vertex passing through vertices that were not already visited. If thiscannot be done, move back another vertex in the path, that is, two vertices back in the path, andtry again.

Repeat this procedure, beginning at the last vertex visited, moving back up the path onevertex at a time, forming new paths that are as long as possible until no more edges can beadded. Because the graph has a finite number of edges and is connected, this process ends withthe production of a spanning tree. Each vertex that ends a path at a stage of the algorithm willbe a leaf in the rooted tree, and each vertex where a path is constructed starting at this vertexwill be an internal vertex.

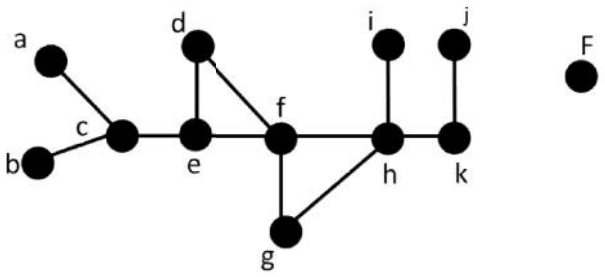


Figure 6 The graph G

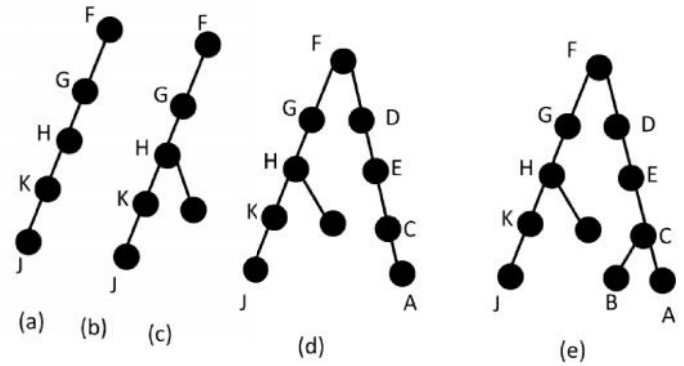


Figure 7 Depth –first search of G

The reader should note the recursive nature of this procedure. Also, note that if the verticesin the graph are ordered, the choices of edges at each stage of the procedure are all determinedwhen we always choose the first vertex in the ordering that is available. However, we will notalways explicitly order the vertices of a graph.

Depth-first search is also called backtracking, because the algorithm returns to verticespreviously visited to add paths. Example 3 illustrates backtracking.

2. Use depth-first search to find a spanning tree for the graph G shown in Figure 6.

Solution: The steps used by depth-first search to produce a spanning tree of G are shown inFigure 7. We arbitrarily start with the vertex f . A path is built by successively adding edgesincident with vertices not already in the path, as long as this is possible. This produces a pathf , g, h, k, j (note that other paths could have been built). Next, backtrack to k. There is nopath beginning at k containing vertices not already visited. So, we backtrack to h. Form thepath h, i. Then backtrack to h, and then to f. From f build the path f, d, e, c, a. Then backtrackto c and form the path c, b. This produces the spanning tree.

The edges selected by depth-first search of a graph are called tree edges. All other edgesof the graph must connect a vertex to an ancestor or descendant of this vertex in the tree. Theseedges are called back edges. (Exercise 43 asks for a proof of this fact.)

3. In Figure 8 we highlight the tree edges found by depth-first search starting at vertex f by showing them with heavy colored lines. The back edges (e, f) and (f, h) are shown with thinner backlines.

We have explained how to find a spanning tree of a graph using depth-first search. However,our discussion so far has not brought out the recursive nature of depth-first search. To helpmake the recursive nature of the algorithm clear, we need a little terminology. We say that we

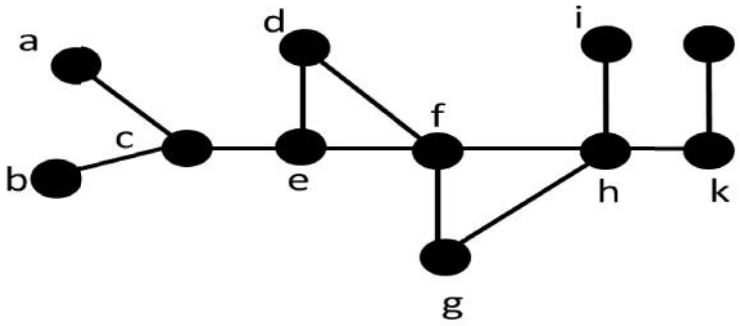


Figure 8 : the tree edges and back edges of the depth fist search in example 4

Explore from a vertex v when we carry out the steps of depth-first search beginning when v isadded to the tree and ending when we have backtracked back to v for the last time. The keyobservation needed to understand the recursive nature of the algorithm is that when we add an edge connecting a vertex v to a vertex w, we finish exploring from w before we return to v tocomplete exploring from v.

In Algorithm 1 we construct the spanning tree of a graph G with vertices v1,..., vn by firstselecting the vertex v1 to be the root. We initially set T to be the tree with just this one vertex.

At each step we add a new vertex to the tree T together with an edge from a vertex alreadyin T to this new vertex and we explore from this new vertex. Note that at the completion ofthe algorithm, T contains no simple circuits because no edge is ever added that connects twovertices in the tree. Moreover, T remains connected as it is built. (These last two observationscan be easily proved via mathematical induction.) Because G is connected, every vertex in Gis visited by the algorithm and is added to the tree (as the reader should verify). It followsthat T is a spanning tree of G.

ALGORITHM 1 Depth-First Search.

procedure DFS (G: connected graph with vertices v_1, v_2,..., v_n)

T: = tree consisting only of the vertex v1

visit(v_1)

procedure visit (v: vertex of G)

for each vertex w adjacent to v and not yet in Tadd vertex w and edge {v, w} to T

visit(w)

We now analyze the computational complexity of the depth-first search algorithm. The keyobservation is that for each vertex v, the procedure visit(v) is called when the vertex v is firstencountered in the search and it is not called again. Assuming that the adjacency lists for G areavailable, no computations are required to find the vertices adjacent to v. Aswe follow the steps of the algorithm, we examine each edge at most twice to determine whetherto add this edge and one of its endpoints to the tree. Consequently, the procedure DFS constructsa spanning tree using O(e), or O(n2), steps where e and n are the number of edges and verticesin G, respectively. [Note that a step involves examining a vertex to see whether it is already inthe spanning tree as it is being built and adding this vertex and the corresponding edge if thevertex is not already in the tree. We have also made use of the inequality $e \leq n(n-1)/2$, whichholds for any simple graph.

Depth-first search can be used as the basis for algorithms that solve many different problems.

For example, it can be used to find paths and circuits in a graph, it can be used to determinethe connected components of a graph, and it can be used to find the cut vertices of a connectedgraph.

13.6 Breadth-First Search

We can also produce a spanning tree of a simple graph by the use of breadth-first search.

Again, a rooted tree will be constructed, and the underlying undirected graph of this rooted tree

forms the spanning tree. Arbitrarily choose a root from the vertices of the graph. Then add all

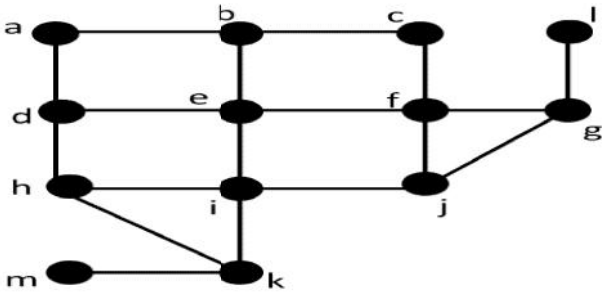


Figure 9 A graph G

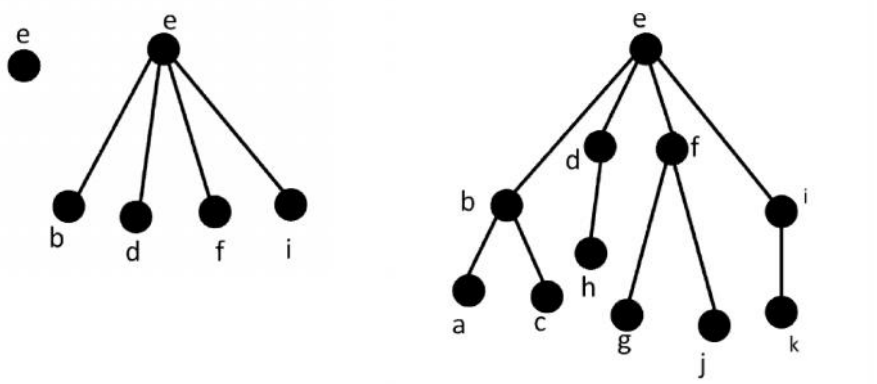
edges incident to this vertex. The new vertices added at this stage become the vertices at level 1in the spanning tree. Arbitrarily order them. Next, for each vertex at level 1, visited in order,add each edge incident to this vertex to the tree as long as it does not produce a simple circuit.

Arbitrarily order the children of each vertex at level 1. This produces the vertices at level 2in the tree. Follow the same procedure until all the vertices in the tree have been added. Theprocedure ends because there are only a finite number of edges in the graph. A spanning tree isproduced because we have produced a tree containing every vertex of the graph. An example ofbreadth-first search is given in Example 5.

5. Use breadth-first search to find a spanning tree for the graph shown in Figure 9.

Solution: The steps of the breadth-first search procedure are shown in Figure 10. We choosethe vertex e to be the root. Then we add edges incident with all vertices adjacent to e, so edgesfrom e to b, d, f , and i are added. These four vertices are at level 1 in the tree. Next, add theedges from these vertices at level 1 to adjacent vertices not already in the tree. Hence, the edgesfrom b to a and c are added, as are edges from d to h, from f to j and g, and from i to k. Thenew vertices a, c, h, j , g, and k are at level 2. Next, add edges from these vertices to adjacentvertices not already in the graph. This adds edges from g to l and from k to m. ▲

We describe breadth-first search in pseudo code asAlgorithm 2. In this algorithm, we as some the vertices of the connected graph G are ordered as v_1, v_2,..., v_n. In the algorithm we use theterm "process" to describe the procedure of adding new vertices, and corresponding edges, tothe tree adjacent to the current vertex being processed as long as a simple circuit is not produced.



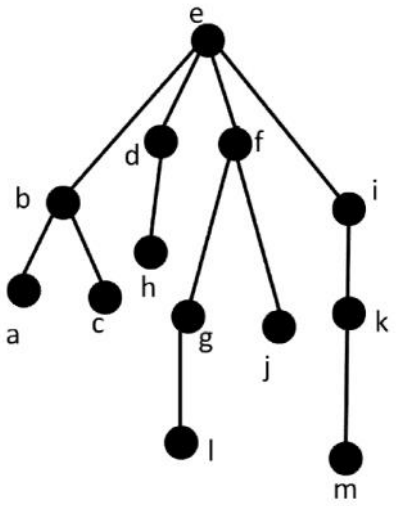


Figure 10 Breadth – first search Of G

ALGORITHM 2 Breadth-First Search.

procedure BFS (G: connected graph with vertices v_1, v_2,..., v_n)

T:= tree consisting only of vertex v1

L:= empty list

put v1 in the list L of unprocessed vertices

while L is not empty

remove the first vertex, v, from L

for each neighbor w of v

if w is not in L and not in T then

add w to the end of the list L

add w and edge {v, w} to T

We now analyze the computational complexity of breadth-first search. For each vertex v in

The graph we examine all vertices adjacent to v and we add each vertex not yet visited to thetree T. Assuming we have the adjacency lists for the graph available, no computation is requiredto determine which vertices are adjacent to a given vertex. As in the analysis of the depth-firstsearch algorithm, we see that we examine each edge at most twice to determine whether weshould add this edge and its endpoint not already in the tree. It follows that the breadth-firstsearch algorithm uses O(e) or O(n2) steps.

Breadth-first search is one of the most useful algorithms in graph theory. In particular it canserve as the basis for algorithms that solve a wide variety of problems. For example, algorithmsthat find the connected components of a graph, that determine whether a graph is bipartite, andthat find the path with the fewest edges between two vertices in a graph can all be built usingbreadth-first search.

Backtracking Applications

There are problems that can be solved only by performing an exhaustive search of all possible

Solutions: One way to search systematically for a solution is to use a decision tree, where eachinternal vertex represents a decision and each leaf a possible solution. To find a solution viabacktracking, first make a sequence of decisions in an attempt to reach a solution as long as this

is possible. The sequence of decisions can be represented by a path in the decision tree. Onceit is known that no solution can result from any further sequence of decisions, backtrack to theparent of the current vertex and work toward a solution with another series of decisions, if this ispossible.

The procedure continues until a solution is found, or it is established that no solutionexists. Examples 6 to 8 illustrate the usefulness of backtracking.

6. Graph Colorings how can backtracking be used to decide whether a graph can be colored using n colors?

Solution: We can solve this problem using backtracking in the following way. First pick somevertex a and assign it color 1. Then pick a second vertex b, and if b is not adjacent to a, assign

it color 1. Otherwise, assign color 2 to b. Then go on to a third vertex c. Use color 1, if possiblefor c. Otherwise use color 2, if this is possible. Only if neither color 1 nor color 2 can be usedshould color 3 be used. Continue this process as long as it is possible to assign one of the ncolors to each additional vertex, always using the first allowable color in the list. If a vertex isreached that cannot be colored by any of the n colors, backtrack to the last assignment made andchange the coloring of the last vertex colored, if possible, using the next allowable color in thelist. If it is not possible to change this coloring, backtrack farther to previous assignments, onestep back at a time, until it is possible to change a coloring of a vertex. Then continue assigning

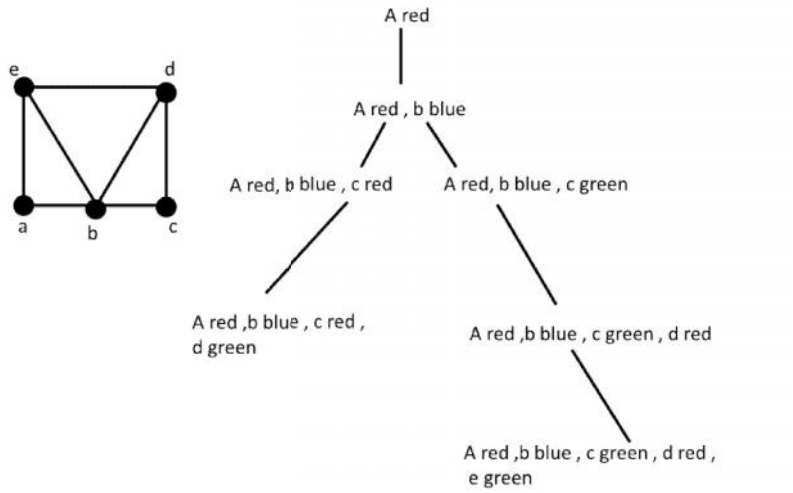


FIGURE 11 coloring a graph using backtracking

colors of additional vertices as long as possible. If a coloring using n colors exists, backtrackingwill produce it. (Unfortunately, this procedure can be extremely inefficient.)In particular, consider the problem of coloring the graph shown in Figure 11 with threecolors. The tree shown in Figure 11 illustrates how backtracking can be used to construct a3-coloring. In this procedure, red is used first, then blue, and finally green. This simple examplecan obviously be done without backtracking, but it is a good illustration of the technique.

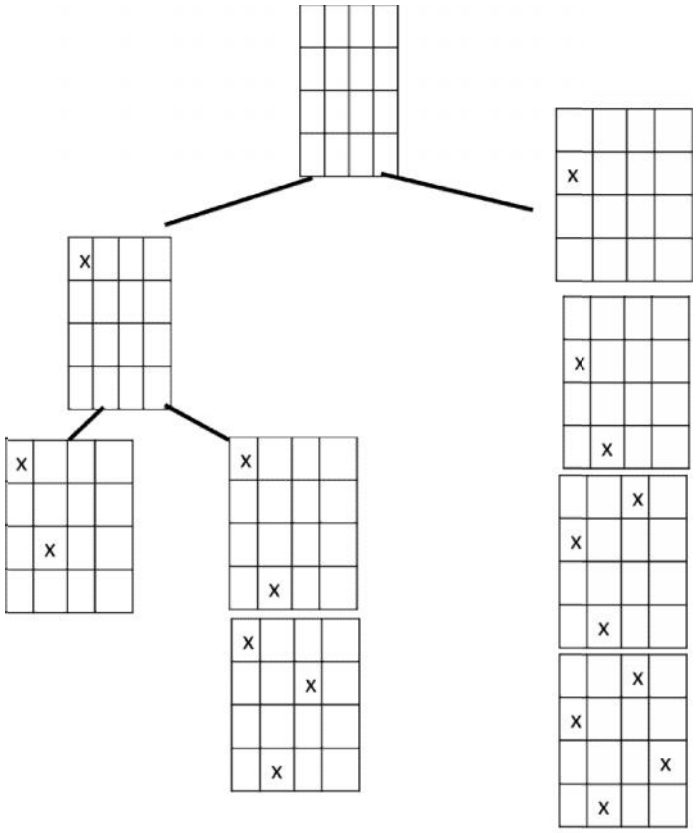
In this tree, the initial path from the root, which represents the assignment of red to a, leadsto a coloring with a red, b blue, c red, and d green. It is impossible to color e using any ofthe three colors when a, b, c, and d are colored in this way. So, backtrack to the parent of thevertex representing this coloring. Because no other color can be used for d, backtrack one morelevel. Then change the color of c to green. We obtain a coloring of the graph by then assigning red to d and green to e. ▲

7. The n-Queens Problem The n-queens problem asks how n queens can be placed on an n × nchessboard so that no two queens can attack one another. How can backtracking be used to solvethe n-queens problem?

Solution: To solve this problem we must find n positions on an n × n chessboard so that notwo of these positions are in the same row, same column, or in the same diagonal [a diagonalconsists of all positions (i, j) with i + j = m for some m, or i − j = m for some m]. We willuse backtracking to solve the n-queens problem. We start with an empty chessboard. At stagek + 1 we attempt putting an additional queen on the board in the (k + 1)st column, where thereare already queens in the first k columns. We examine squares in the (k + 1)st column startingwith the square in the first row, looking for a position to place this queen so that it is not in thesame row or on the same diagonal as a queen already on the board. (We already know it is notin the same column.) If it is impossible to find a position to place the queen in the (k + 1)stcolumn, backtrack to the placement of the queen in

the kth column, and place this queen in thenext allowable row in this column, if such a row exists. If no such row exists, backtrack further.

In particular, Figure 12 displays a backtracking solution to the four-queens problem. In thissolution, we place a queen in the first row and column. Then we put a queen in the third row ofthe second column. However, this makes it impossible to place a queen in the third column. Sowe backtrack and put a queen in the fourth row of the second column. When we do this, we canplace a queen in the second row of the third column. But there is no way to add a queen to thefourth column. This shows that no solution results when a queen is placed in the first row andcolumn. We backtrack to the empty chessboard, and place a queen in the second row of the firstcolumn. This leads to a solution as shown in Figure 12.



X represents queen

Figure 12 A backtracking solution of the four – queen problem.

8. Sums of Subsets Consider this problem. Given a set of positive integers x_1, x_2,...,x_n, finda subset of this set of integers that has M as its sum. How can backtracking be used to solve thisproblem?

Solution: We start with a sum with no terms. We build up the sum by successively adding terms.

An integer in the sequence is included if the sum remains less than M when this integer is added

to the sum. If a sum is reached such that the addition of any term is greater than M, backtrack

by dropping the last term of the sum.

Figure 13 displays a backtracking solution to the problem of finding a subset of{31, 27, 15, 11, 7, 5} with the sum equal to 39.

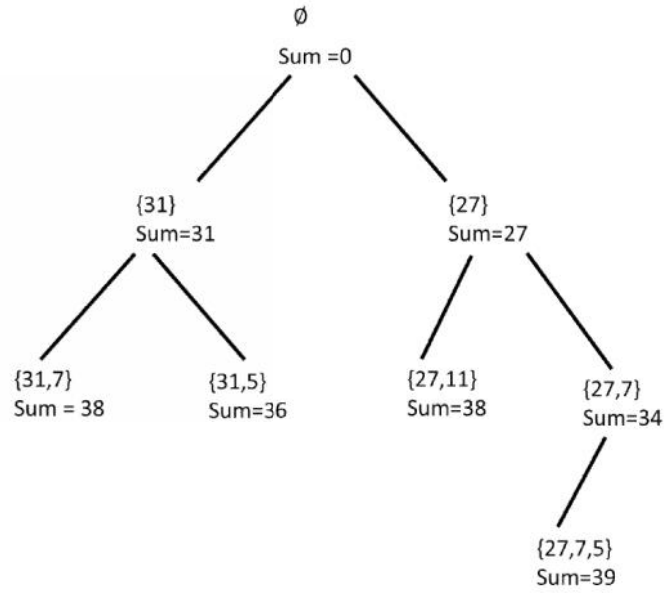


Figure 13 find a sum equal to 39 using backtracking

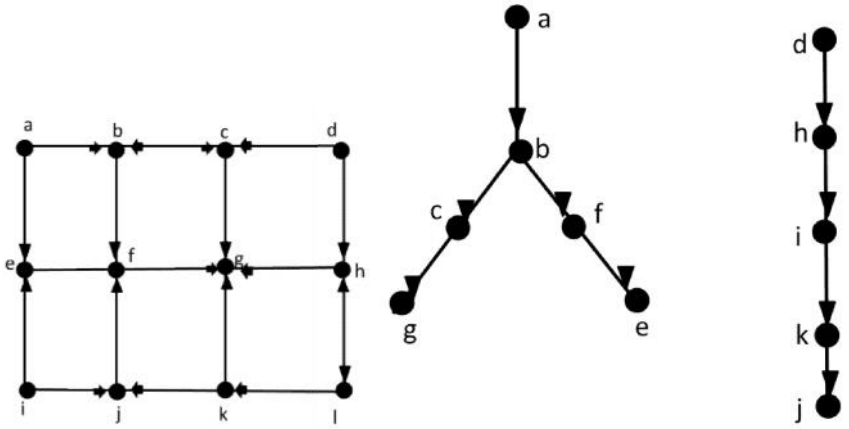


Figure 14 depth- first search of a directed graph

13.7 Depth-First Search in Directed Graphs

We can easily modify both depth-first search and breadth-first search so that they can run given a directed graph as input. However, the output will not necessarily be a spanning tree, but rather a spanning forest. In both algorithms we can add an edge only when it is directed away fromthe vertex that is being visited and to a vertex not yet added. If at a stage of either algorithm wefind that no edge exists starting at a vertex already added to one not yet added, the next vertex added by the algorithm becomes the root of a new tree in the spanning forest. This is illustrated in Example 9.

9. What is the output of depth-first search given the graph G shown in Figure 14(a) as input?

Solution: We begin the depth-first search at vertex a and add vertices b, c, and g and thecorresponding edges where we are blocked. We backtrack to c but we are still blocked, and then backtrack to b, where we add vertices f and e and the corresponding edges. Backtracking takes us all the way back to a. We then start a new tree at d and add vertices h, l, k, and j and thecorresponding edges. We backtrack to k, then l, then h, and back to d. Finally, we start a new tree at i, completing the depth-first search. The output is shown in Figure 14(b). ▲

Depth-first search in directed graphs is the basis of many algorithms. It can be used to determine whether a directed graph has a circuit, it canbe used to carry out a topological sort of a graph, and it can also be used to find the stronglyconnected components of a directed graph.

We conclude this section with an application of depth-first search and breadth-first searchto search engines on the Web.

10. Web Spiders To index websites, search engines such as Google and Yahoo systema bit call explore the Web starting at known sites. These search engines use programs called Web spiders(or crawlers or bots) to visit websites and analyze their contents. Web spiders use both depth-firstsearching and breadth-first searching to create indices .Asdescribed; Web pages and links between them can be modeled by a directed graph called the Web graph.

Web pages are represented by vertices and links are represented by directed edges. Using depthfirst search, an initial Web page is selected; a link is followed to a second Web page (if there is

such a link), a link on the second Web page is followed to a third Web page, if there is such alink, and so on, until a page with no new links is found. Backtracking is then used to examine 11.4 Spanning Trees 795links at the previous level to look for new links, and so on. (Because of practical limitations, Webspiders have limits to the depth they search in depth-first search.) Using breadth-first search, aninitial Web page is selected and a link on this page is followed to a second Web page, then asecond link on the initial page is followed (if it exists), and so on, until all links of the initialpage have been followed. Then links on the pages one level down are followed, page by page,and so on.

13.8 Algorithms for Minimum Spanning Trees

11. A company plans to build a communications network connecting its five computer centers. Any pair of these centers can be linked with a leased telephone line. Which links should be made to ensure that there is a path between any two computer centers so that the total cost of the network is minimized? We can model this problem using the weighted graph shown in Figure 1, where vertices represent computer centers, edges represent possible leased lines, and the weights on edges are the monthly lease rates of the lines represented by the edges. We can solve this problem

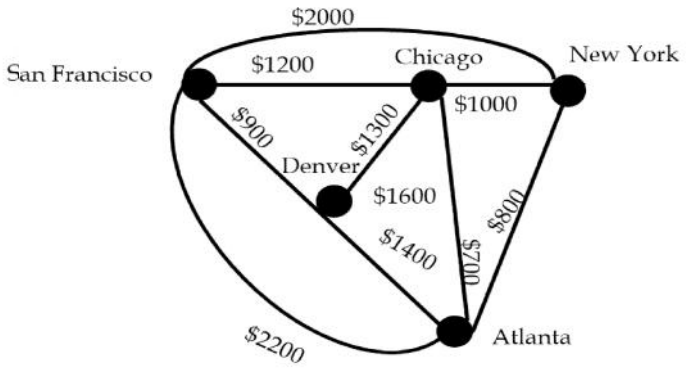


Figure 1 A Weighted Graph Showing Monthly Lease Costs for Lines in a Computer Network.

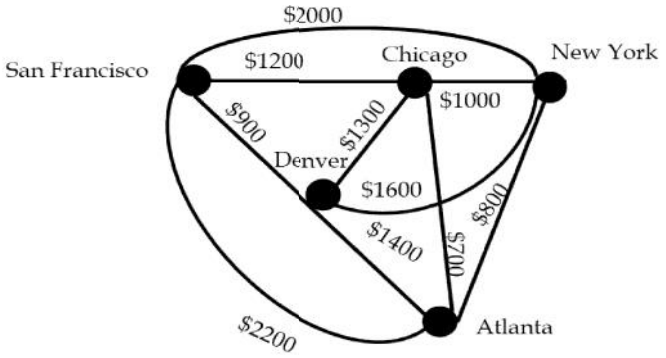
by finding a spanning tree so that the sum of the weights of the edges of the tree is minimized. Such a spanning tree is called a minimum spanning tree.

A wide variety of problems are solved by finding a spanning tree in a weighted graph such that the sum of the weights of the edges in the tree is a minimum.

DEFINITION 1 A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

We will present two algorithms for constructing minimum spanning trees. Both proceed by successively adding edges of smallest weight from those edges with a specified property that have not already been used. Both are greedy algorithms. Recal that a greedy algorithm is a procedure that makes an optimal choice a teach of it steps. Optimizing at each step does not guarantee that the optimal overall solution is produced. However, the two algorithms presented in this section for constructing minimum spanning trees are greedy algorithms that do produce optimal solutions.

The first algorithm that we will discuss was originally discovered by the Czech mathematician VojteˇchJarník in 1930, who described it in a paper in an obscure Czech journal. The algorithm became well known when it was rediscovered in 1957 by Robert Prim. Because of this, it is known as Prim's algorithm (and sometimes as the Prim-Jarník algorithm). Begin by choosing any edge with smallest weight, putting it into the spanning tree. Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree, never forming a simple circuit with those edges already in the tree. Stop when n−1 edges have been added. Later in thissection, we will prove that this algorithm produces a minimum spanning tree for any connected weighted graph. Algorithm1 gives apseudocodedescriptionof Prim's algorithm.



Choice	edges	Cost
1	{Chicago, Atlanta	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$ 1200
4	{San Francisco, Denver}	$ 900
		Total $3600

FIGURE 2 A Minimum Spanning Tree for the Weighted Graph in Figure 1.

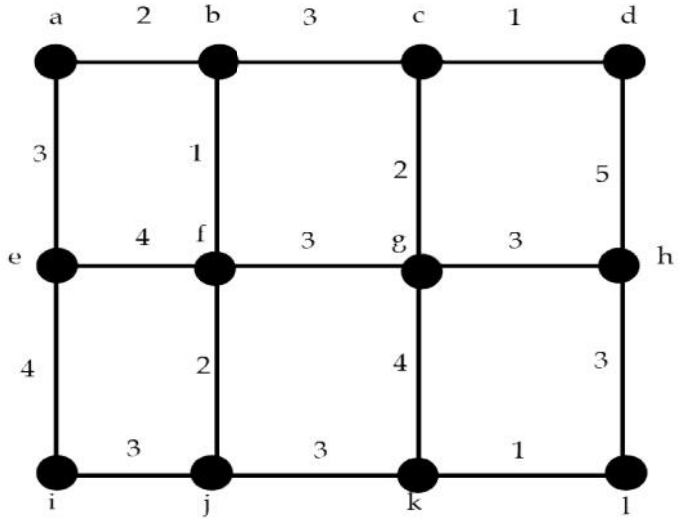


Figure 3 A Weighted Graph.

ALGORITHM 1 Prim's Algorithm.

Procedure Prim (G: weighted connected undirected graph with n vertices)

T: = a minimum-weight edge

for i: = 1 to n−2

e: =an edge of minimum weight incident to a vertex in T and not forming asimple circuit in T if added to T

T: = T with e added

Return T {T is a minimum spanning tree of G}

Note that the choice of an edge to add at a stage of the algorithm is not determined when there is more than one edge with the same weight that satisfies the appropriate criteria. We need to order the edges to make the choices deterministic. We will not worry about this in the remainder of the section. Also note that there may be more than one minimum spanning tree for a given connected weighted simple graph. Examples 1 and 2 illustrate how Prim's algorithm is used.

1. Use Prim's algorithm to design a minimum-cost communications network connecting all the computers represented by the graph in Figure 1.

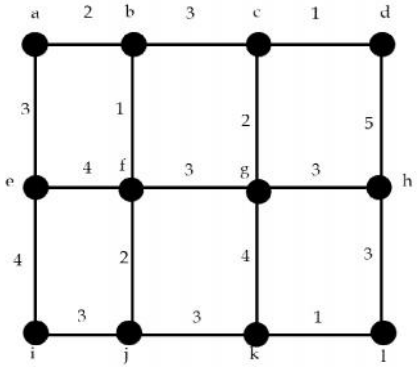
Solution: We solve this problem by finding a minimum spanning tree in the graph in Figure 1. Prim's algorithm is carried out by choosing an initial edge of minimum weight and successively adding edges of minimum weight that are incident to a vertex in the tree and that do not form simple circuits. The edges in color in Figure 2 show a minimum spanning tree produced by Prim's algorithm, with the choice made at each step displayed.

▲

2. Use Prim's algorithm to find a minimum spanning tree in the graph shown in Figure 3.

Solution: A minimum spanning tree constructed using Prim's algorithm is shown in Figure 4. The successive edges chosen are displayed. ▲

The second algorithm we will discuss was discovered by Joseph Kruskal in 1956, although the basic ideas it uses were described much earlier. To carry out Kruskal's algorithm, choose an edge in the graph with minimum weight.



Choice	Edge	weight
1	{ b, f}	1
2	{ a, b}	2
3	{ F, j }	2
4	{ a, e}	3
5	{ i, f }	3
6	{ f , g}	3
7	{c , g}	2
8	{c, d}	1
9	{ g ,h }	3
10	{h ,l }	13
11	{ k ,l}	1
		Total 24

FIGURE 4 A Minimum Spanning Tree Produced Using Prim's Algorithm.

Successively add edges with minimum weight that do not form a simple circuit with those edges already chosen. Stop after n−1 edges have been selected. The proof that Kruskal's algorithm produces

a minimum spanning tree for every connected weighted graph is leftasan exercise. Pseudo code for Kruskal's algorithmis givenin Algorithm2

ALGORITHM 2 Kruskal's Algorithm.

Procedure Kruskal (G: weighted connected undirected graph with n vertices)

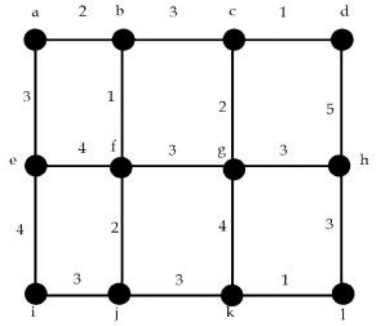
T: = empty graph

for i: = 1 to n−1

e: =any edge in G with smallest weight that does not form a simple circuit when added to T

T: = T with e added

return T {T is a minimum spanning tree of G}



Choice	Edge	weight
1	{ c, d}	1
2	{ k, l}	1
3	{ b, f }	1
4	{ c, g}	2
5	{ a, b}	2
6	{ f , j}	2
7	{b , c}	3
8	{j, k}	3
9	{ g ,h }	3
10	{i ,j }	3
11	{ a ,e}	3
		Total 24

FIGURE 5 A Minimum Spanning Tree Produced by Kruskal's Algorithm.

The reader should note the difference between Prim's and Kruskal's algorithms. In Prim's algorithm edges of minimum weight that are incident to a vertex already in the tree, and not forming a circuit, are chosen; whereas in Kruskal's algorithm edges of minimum weight that are not necessarily incident to a vertex already in the tree, and that do not form a circuit, are chosen. Note that as in Prim's algorithm, if the edges are not ordered, there may be more than one choice for the edge to add at a stage of this procedure. Consequently, the edges need to be ordered for the procedure to be deterministic. Example 3 illustrates how Kruskal's algorithm is used.

3. Use Kruskal's algorithm to find a minimum spanning tree in the weighted graph shown in Figure 3.

Solution: A minimum spanning tree and the choices of edges a teach stage of Kruskal's algorithm are shown in Figure 5.

We will now prove that Prim's algorithm produces a minimum spanning tree of a connected weighted graph.

Proof: Let G be a connected weighted graph. Suppose that the successive edges chosen by Prim's algorithm are e_1, e_2..., e_{n-1}. Let S be the tree with e_1, e_2..., e_{n-1} as its edges, and let S_k be the tree with e_1, e_2..., e_k as its edges. Let T be a minimum spanning tree of G containing the edges e_1, e_2..., e_k, where k is the maximum integer with the property that a minimum spanning tree exists containing the first k edges chosen by Prim's algorithm. The theorem follows if we can show that S = T.

Suppose that S ≠= T, so that k<n−1. Consequently, T contains e_1, e_2..., e_k, but not e_{k+1}. Consider the graph made up of T together with e_{k+1}. Because this graph is connected and has n edges, too many edges to be a tree, it must contain a simple circuit. This simple circuit must contain ek+1 because there was no simple circuit in T. Furthermore, there must be an edge in the simple circuit that does not belong to S_{k+1} because S_{k+1} is a tree. By starting at an endpoint of e_{k+1} that is also an endpoint of one of the edges e_1,...,e_k, and following the circuit until it reaches an edge not in S_{k+1}, we can find an edge e not in S_{k+1} that has an endpoint that is also an endpoint of one of the edges e_1,e_2,...,e_k. By deleting e from T and adding e_{k+1}, we obtain a tree T′ with n−1 edges (it is a tree because it has no simple circuits). Note that the tree T′ contains e_1, e_2..., e_k, e_{k+1}. Furthermore, because e_{k+1} was chosen by Prim's algorithm at the $(k+1)^{st}$ step, and e was also available at that step, the weight of e_{k+1} is less than or equal to the weight of e. From this observation, it follows that T′ is also a minimum spanning tree, because the sum of the weights of its edges does not exceed the sum of the weights of the edges of T. This contradicts the choice of k as the maximum integer such that a minimum spanning tree exists containing e_1..., e_k. Hence, k = n−1, and S = T. It follows that Prim's algorithm produces a minimum spanning tree. It can be shown that to find a minimum spanning tree of a graph with m edges and n vertices, Kruskal's algorithm can be carried out using O(mlogm) operations and Prim's algorithm can be carried out using O(mlogn) operations. Consequently, it is preferable to use Kruskal's algorithm for graphs that are sparse, that is, where m is very small compared to C(n,2) = n(n−1)/2, the total number of possible edges in a nun directed graph withinvertices. Otherwise, there is little difference in the complexity of these two algorithms.

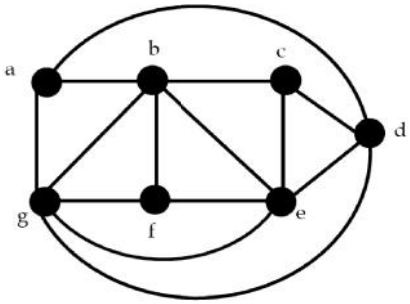
Summary

- We have learned what are Spanning Trees
- We have learned what is Kruskal's Algorithm.
- We have learned what Prim's Algorithm is.

Self Assessment

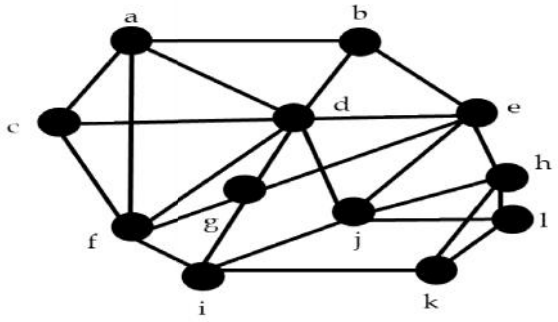
1. How many edges must be removed from a connected graph with n vertices and m edges to produce a spanning tree?
   A. The graph has m edges. The spanning tree has n-2 edges. Therefore, we need to remove m-(n-2) edges.
   B. The graph has m edges. The spanning tree has n-3edges. Therefore, we need to remove m-(n-3) edges.
   C. The graph has m edges. The spanning tree has n-1 edges. Therefore, we need to remove m-(n-1) edges.
   D. The graph has m edges. The spanning tree has n-4 edges. Therefore, we need to remove m-(n-4) edges.

2. Find a spanning tree for the graph shown by removing edges in simple circuits.



   A. We have to remove edges, one at a time. We can remove any edge that is part of a simple circuit. The answer is by no means unique. For example, we can start by removing edge {a, c}, since it is in the simple circuit adcba. Then we might choose to remove edge {a, g}. We can continue in this way and remove all of these edges: {b, e}, {b, a}, {b, d}, {d, e}, {r, g}, and {k, g}. At this point there are no more simple circuits, so we have a spanning tree.
   B. We have to remove edges, one at a time. We can remove any edge that is part of a simple circuit. The answer is by no means unique. For example, we can start by removing edge {a, d}, since it is in the simple circuit adcba. Then we might choose to remove edge {a, g}. We can continue in this way and remove all of these edges: {b, e}, {b, J}, {b, g}, {d, e}, {d, g}, and {e, g}. At this point there are no more simple circuits, so we have a spanning tree.
   C. We have to remove edges, one at a time. We can remove any edge that is part of a simple circuit. The answer is by no means unique. For example, we can start by removing edge {a, d}, since it is in the simple circuit adcba. Then we might choose to remove edge {a, g}. We can continue in this way and remove all of these edges: {e, e}, {b, m}, {b, n}, {d, m}, {n, g}, and {l, g}. At this point there are no more simple circuits, so we have a spanning tree.
   D. We have to remove edges, one at a time. We can remove any edge that is part of a simple circuit. The answer is by no means unique. For example, we can start by removing edge {a, d}, since it is in the simple circuit adcba. Then we might choose to remove edge {a, g}. We can continue in this way and remove all of these edges: {b, c}, {d, J}, {g, g}, {g, e}, {h, g}, and {h, g}. At this point there are no more simple circuits, so we have a spanning tree

3. Find a spanning tree for the graph shown by removing edges in simple circuits.



A. Here isonepossiblesetofremovals: {a,b}, {a,d}, {a,!}, {b,c}, {b,d}, {c,d}, {d,e}, {d,g}, {d,j}, {e,g}, {e,j}, {f,g}, {h,j}, {h,k}, and {j,l}. As a check, note that there are 12 vertices and 26 edges. A spanning tree must have 11 edges, so we need to remove 15 of them, as we did.

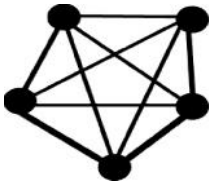
B. Hereisonepossiblesetofremovals: {a,b}, {a,d}, {a,!}, {b,c}, {c,d}, {c,e}, {f,e}, {g,g}, {h,j}, {e,i}, {j,j}, {f,l}, {h,m}, {h,k}, and {j,c}. As a check, note that there are 12 vertices and 26 edges. A spanning tree must have 11 edges, so we need to remove 15 of them, as we did.

C. Hereisonepossiblesetofremovals: {a,b}, {a,d}, {a,!}, {b,c}, {b,d}, {c,b}, {d,n}, {d,l}, {j,j}, {k,g}, {l,j}, {m,g}, {0,j}, {g,k}, and {s,l}. As a check, note that there are 12 vertices and 39 edges. A spanning tree must have 11 edges, so we need to remove 15 of them, as we did.

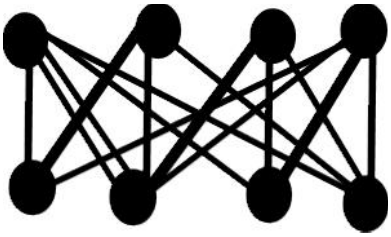
D. Hereisonepossiblesetofremovals: {a,b}, {a,d}, {a,!}, {b,c}, {b,d}, {c,d}, {d,e}, {d,g}, {d,j}, {e,g}, {e,j}, {f,g}, {h,j}, {h,k}, and {j,l}. As a check, note that there are 78 vertices and 66 edges. A spanning tree must have 15 edges, so we need to remove 09 of them, as we did.

4. **A** spanning tree for the graph$K_5$is

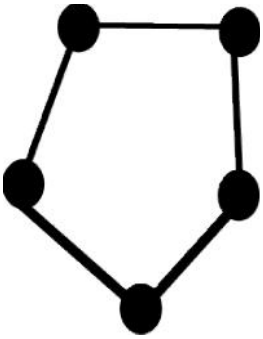
A.



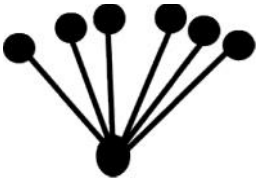
B.



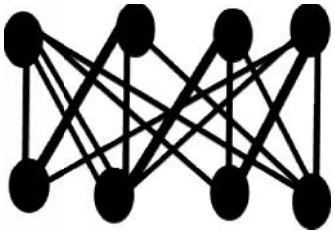
C.

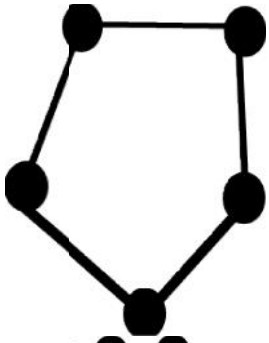


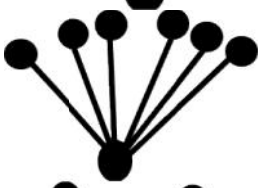
D.

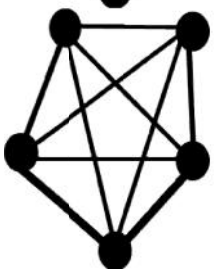


5. A spanning tree for the graph $K_{4,4}$is

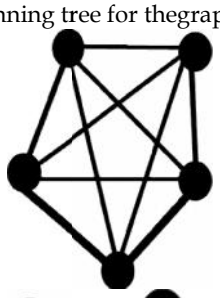
A. 

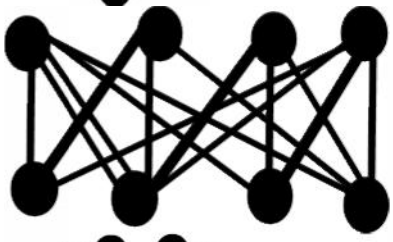
B. 

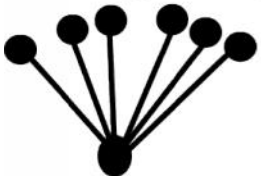
C. 

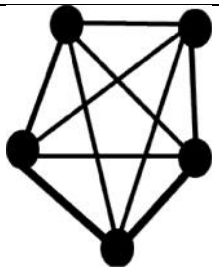
D. 

6. A spanning tree for thegraph $K_{1,6}$is

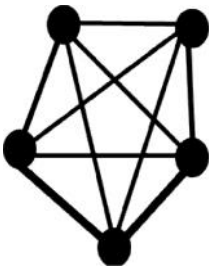
A. 

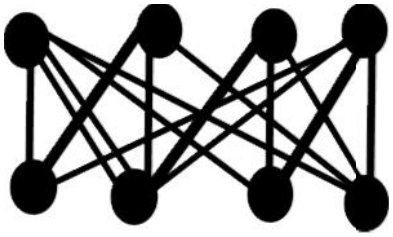
B. 

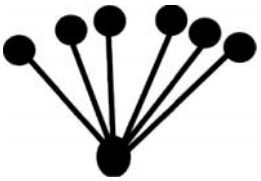
C. 

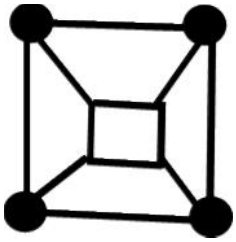
D. 

7. A spanning tree for the graph $Q_3$is

A. 

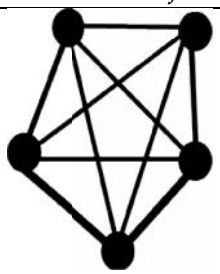
B. 

C. 

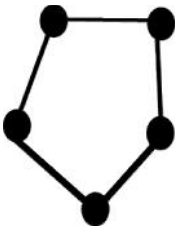
D. 

8. A spanning tree for the graph $C_5$is

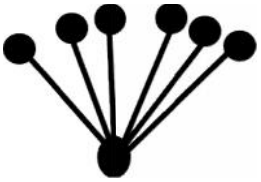
A.



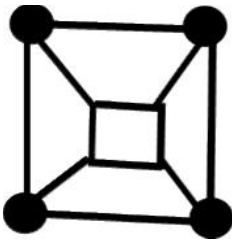
B.



C.

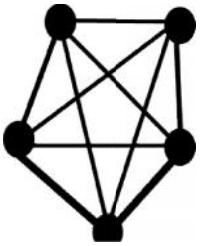


D.

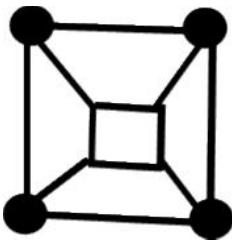


9. A spanning tree for the graph $W_5$is

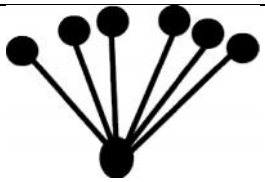
A.



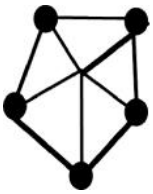
B.

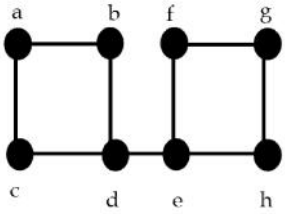


C.



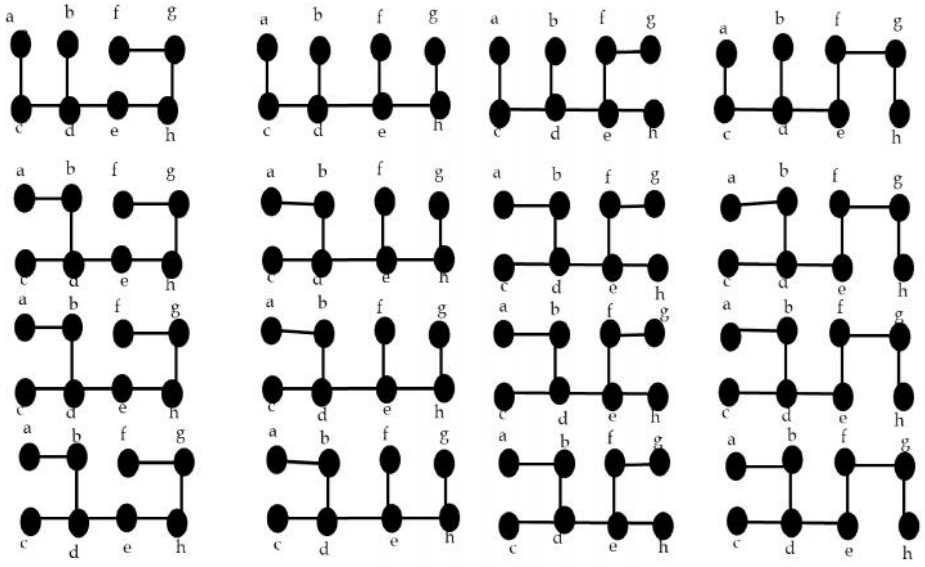
D.



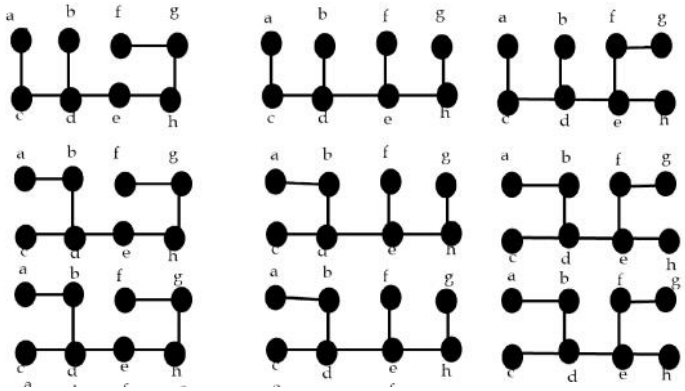
10. All the spanning trees of the given simple graph 

are

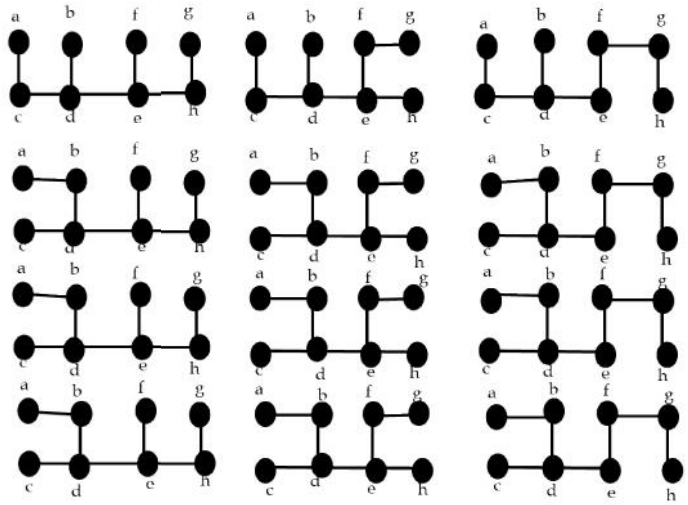
A.



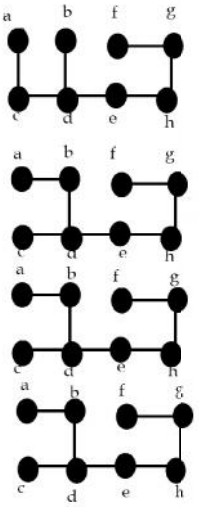
B.



C.



D.



11. How many different spanning trees does the simple graphK_3have?

A. Every pair of edges in K_3 forms a spanning tree, so there are C (3, 3) = 1 such trees.
B. Every pair of edges in K_3 forms a spanning tree, so there are C (2, 2) = 1 such trees.
C. Every pair of edges in K_3 forms a spanning tree, so there are C (4, 2) = 6 such trees.
D. Every pair of edges in K_3 forms a spanning tree, so there are C (3, 2) = 3 such trees

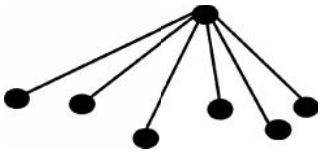
12. How many different spanning trees does the simple graphK_4 have?

A. There are 18 spanning trees
B. There are 16 spanning trees
C. There are 20 spanning trees
D. None of these

13. How many different spanning trees does the simple graph$K_{2,2}$ have?
A. Note that $K_{2,2} = C_4$. A tree is determined simply by deciding which of the four edges to remove. Therefore, there are 8 spanning trees.

B. Note that $K_{2,2} = C_4$. A tree is determined simply by deciding which of the four edges to remove. Therefore, there are 12 spanning trees.
C. Note that $K_{2,2} = C_4$. A tree is determined simply by deciding which of the four edges to remove. Therefore, there are 18 spanning trees.
D. Note that $K_{2,2} = C_4$. A tree is determined simply by deciding which of the four edges to remove. Therefore, there are 4 spanning trees.

14. How many different spanning trees does the simple graphC_5have?
A. there are 15 spanning trees
B. there are 5 spanning trees
C. there are 35 spanning trees
D. there are 75 spanning trees

15. A spanning tree  is for the graph

A. $K_{1,4}$
B. $K_{2,6}$
C. $K_{5,6}$
D. $K_{1,6}$

Answer for Self Assessment

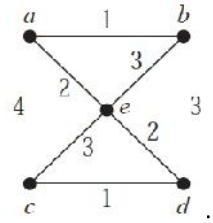
1.	C	2.	B	3.	A	4.	A	5.	A
6.	C	7.	D	8.	B	9.	D	10.	A
11.	D	12.	D	13.	D	14.	B	15.	D

Review Question

1. How many no isomorphic spanning trees does each ofthese simple graphs have?
a) K_3 b) K_4 c) K_5

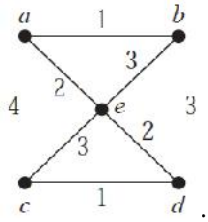
2. Use breadth-first search to find a spanning tree of each ofthe graphs in
a) W_6, starting at the vertex of degree 6
b) K_5
c) $K_{3,4}$, starting at a vertex of degree 3
d) Q_3.

3. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph

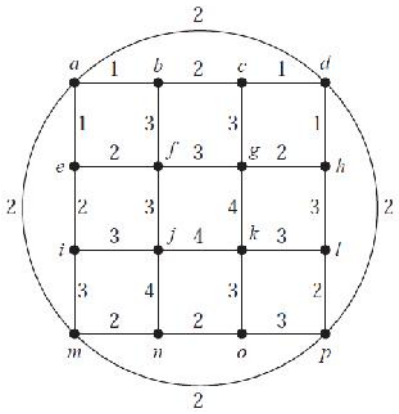


.

4. Use Prim's algorithm to find a minimum spanning tree for the given weighted graph



5. Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph



Further Reading

Rosen, Kenneth H. "Discrete Mathematics and Its Applications."

Rosen, Kenneth H., and Kamala Krithivasan. Discrete mathematics and its applications: with combinatorics and graph theory. Tata McGraw-Hill Education, 2012.

Koshy, Thomas. Discrete mathematics with applications. Elsevier, 2004.

Lipschutz, Seymour, and Marc Lipson. "Schaum's outline of theory and problems of discrete mathematics." (1997).

Unit 14: Tree Traversal

CONTENTS

Objectives

Introduction

14.1 Traversing Binary Trees

14.2 Tree Searching

14.3 Preorder Search Method

14.4 Traversal Algorithms

14.5 Infix, Prefix, and Postfix Notation

Summary

Self Assessment

Answers for Self Assessment

Review Questions

Further Readings

Objectives

The key concepts explained in this unit, including area a learner is expected to learn and master after going through the unit are to: -

- Understand what is tree traversal.
- Understand what arethreetraversal which are called preorder, inorder, and postorder.
- Understand what areInfix, Prefix, and Postfix Notation.

Introduction

Ordered rooted trees are often used to store information. We need procedures for visiting each vertex of an ordered rooted tree to access data. We will describe several important algorithms for visiting all the vertices of an ordered rooted tree. Ordered rooted trees can also be used to represent various types of expressions, such as arithmetic expressions involving numbers, variables, and operations. The different listings of the vertices of ordered rooted trees used to represent expressions are useful in the evaluation of these expressions.

Universal Address Systems

Procedures for traversing all vertices of an ordered rooted tree rely on the orderings of children. In ordered rooted trees, the children of an internal vertex are shown from left to right in the drawings representing these directed graphs. We will describe one way we can totally order the vertices of an ordered rooted tree. To produce this ordering, we must first label all the vertices. We do this recursively:

1. Label the root with the integer 0. Then label its k children (at level 1) from left to right with 1,2, 3, k. 2. For each vertex v at level n with label A, label its k_v children, as they are drawn from left to right, with $A.1$, $A.2$, A. k. $_v$.

Following this procedure, a vertex v at level n, for $n \geq 1$, is labeled $x_1. x_2....x_n$, where the unique path from the root to v goes through the x_1^{st} vertex at level 1, the x_2^{nd} vertex at level 2, and so on. This labeling is called the universal address system of the ordered rooted tree. Wecantotallyordertheverticesusingthelexicographicorderingoftheirlabelsintheuniversal address system. The vertex labeled $x_1. x_2.....x_n$ is less than the vertex labeled $y_1. y_2.....y_m$ if there is an i, $0 \leq i \leq n$, with $x_1 = y_1$, $x_2 = y_2,...,x_{i-1} = y_{i-1}$, and $x_i < y_i$; or if $n<m$ and $x_i = y_i$ for $i = 1,2,...,n$.

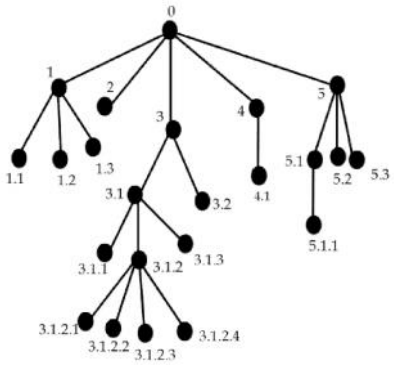


Figure1: The Universal Address System of an Ordered Rooted Tree.

: We display the libeling of the universal address system next to the vertices in the ordered rooted tree shown in Figure 1. The lexicographic ordering of the labelingsis

0 < 1 < 1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 < 3.1.2.3 < 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.1.1 < 5.2 < 5.3 ▲

14.1 Traversing Binary Trees

There are three standard ways of traversing a binary tree T with root R. These three algorithms, called preorder, inorder, and postorder, are as follows:

Preorder: (1) Process the root R.

(2) Traverse the left subtree of R in preorder.

(3) Traverse the right subtree of R in preorder.

Inorder: (1) Traverse the left subtree of R in inorder.

(2) Process the root R.

(3) Traverse the right subtree of R in inorder.

Postorder: (1) Traverse the left subtree of R in postorder.

(2) Traverse the right subtree of R in postorder.

(3) Process the root R.

Observe that each algorithm contains the same three steps, and that the left subtree of R is always traversed before the right subtree. The difference between the algorithms is the time at which the root R is processed. Specifically, in the "pre" algorithm, the root R is processed before the subtrees are traversed; in the "in" algorithm, the root R is processed between the traversals of the subtrees; and in the "post" algorithm, the root R is processed after the subtrees are traversed.

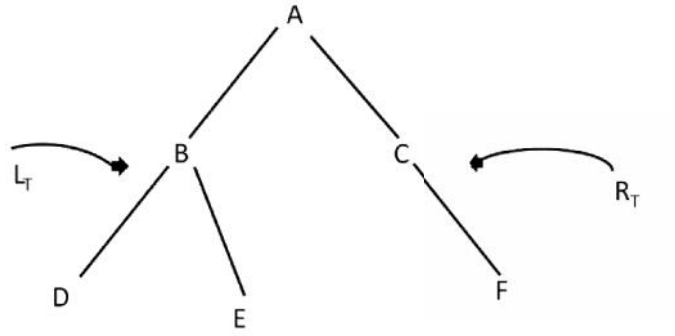
The three algorithms are sometimes called, respectively, the node-left-right (NLR) traversal, the left-node-right (LNR) traversal, and the left-right-node (LRN) traversal.

Consider the binary tree T in Fig. (a). Observe that A is the root of T, that the left subtree L_T of T consists of nodes B, D, and E, and the right subtree R_T of T consists of nodes C and F.

(a) The preorder traversal of T processes A, traverses LT , and traverses R_T . However, the preorder traversal of L_T processes the root B, and then D and E; and the preorder traversal of R_T processes the root C and then F. Thus ABDECF is the preorder traversal of T.

(b) The inorder traversal of T traverses L_T processes A, and traverses R_T . However, the inorder traversal of L_T processes D, B, and then E; and the inorder traversal of R_T processes C and then F. Thus DBEACF is the inorder traversal of T.

(c) The postorder traversal of T traverses L_T , traverses R_T , and processes A. However, the postorder traversal of L_T , processes D, E, and then B, and the postorder traversal of R_T processes F and then C. Accordingly, DEBFCA is the postorder traversal of T.



(a)

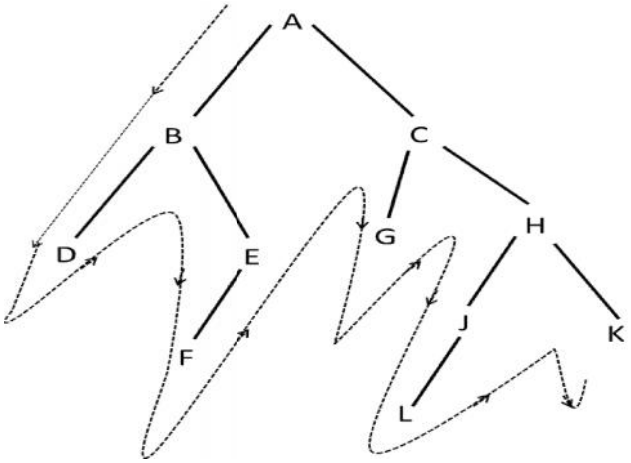


Figure (b)

: LetT be the binary tree in Fig. 10-7(b). The preorder traversal is as follows:

(Preorder)A B D E F C G H J L K This order is the same as the one obtained by scanning the tree from the left as indicated by the path in

Fig. 10-7(b). That is, one "travels" down the leftmost branch until meeting a terminal node, then one backtracks to the next branch, and so on. In the preorder traversal, the rightmost terminal node, node K, is the last node scanned. Observe that the left subtree of the root A is traversed before the right subtree, and both are traversed after A. The same is true for any other node having subtrees, which is the underlying property of a preorder traversal.

The reader can verify by inspection that the other two ways of traversing the tree T in Fig. (b) are as follows:

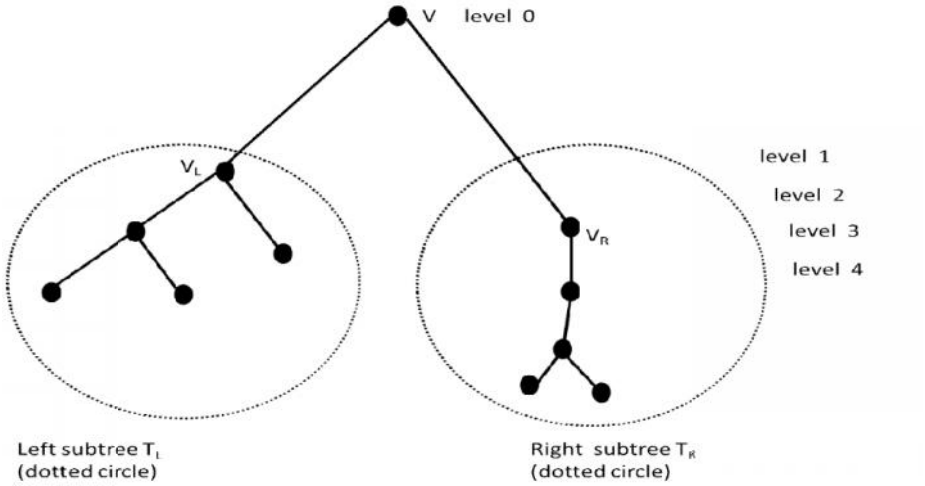
(Inorder) D B F E A G C L J H K

(Postorder)D F E B G L J K H C A

Remark: Observe that the terminal nodes, D, F, G, L, and K, of the binary tree in Fig. 10-7(b) are traversed in the same order, from left to right, in all three traversals. We emphasize that this is true for any binary tree T.

14.2 Tree Searching

Let T be a binary tree of height $h \geq 1$ and root v. Since $h \geq 1$, v has at least one child :v_L and / or v_R. Now $v_L$and $v_R$are the roots of the left and right subtrees of v called T_L and T_R respectively.



Definition: Performing appropriate tasks at a vertex is called visiting the vertex.

Definition: The process of visiting each vertex of a tree in some specified order is called searching the tree or walking or traversing the tree.

We now discuss methods of searching a tree.

14.3 Preorder Search Method

Input: the root v of a binary tree.

Output: Vertices of a binary tree using pre-order traversal

1. Visit v

2. If v_L (left child of v) exists, then apply the algorithm to $(T(v_L), v_L)$

3. If v_R (right child of v) exists, then apply this algorithm to $(T(v_R), v_R)$.

End of Algorithm preorder.

In other words, preorder search of a tree consists of the following steps:

Step 1. Visit the root

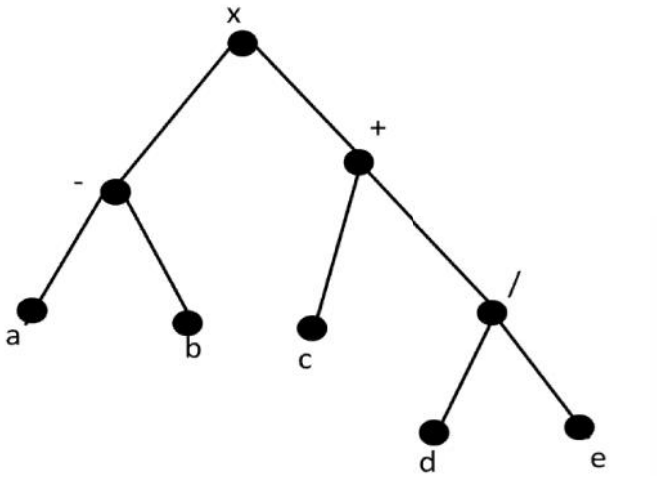
Step 2. Search the left subtree if it exists

Step 3. Search the right subtree if it exists.

Question 1: Find binary tree representation of the expression

$(a - b) \times (c + (d \div e))$ and represent the expression in string form using pre-order traversal.

Solution: In the given expression, X is the central operator and therefore shall be the root of the binary tree. Then the operator – acts as v_L and the operator + acts as v_R. Thus the tree representation of the given expression is



The result of the pre-order traversal to this binary tree is the string

x – a b + c ÷ d e

This form of the expression is called prefix form or polish form of the expression

(a – b) × (c + (d ÷ e))

In a polish form, the variables a, b, c,…are called operands and –, +, ×, ÷ are called operators. We observe that, in polish form, the operands follow the operator.

Procedure to Evaluate an Expression Given Inpolish Form

To find the value of a polish form, we proceed as follows: Move from left to right until we find a string of the form K x y, where K is operator and x, y are operands.

Evaluate x K y and substitute the answer for the string K x y. Continue this procedure until only one number remains.

Question: Find parenthesized form of the polish expression – + A B C

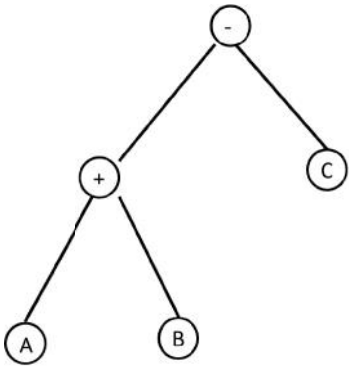
Solution: The parenthesized form of the given polish expression is derived a

follows:

– (A + B) C

(A + B) – C

The corresponding binary tree is



Postorder Search Method

Algorithm

Step 1. Search the left subtree if it exists

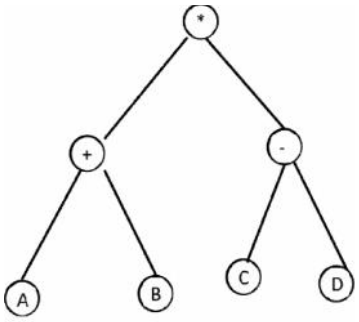
Step 2. Search the right subtree if it exists

Step 3. Visit the root

End of algorithm

Question: Represent the expression (A + B) * (C – D) as a binary tree and write the result of postorder search for that tree.

Solution: The binary tree expression (as shown earlier) of the given algebraic expression is



The result of postorder search of this tree is

A B + C D – *

This form of the expression is called postfix form of the expression or reverse polish form of the expression. In postfix form, the operator follows its operands.

Question: Find the parenthesized form of the postfix form

A B C * * C D E + / –

Solution: We have

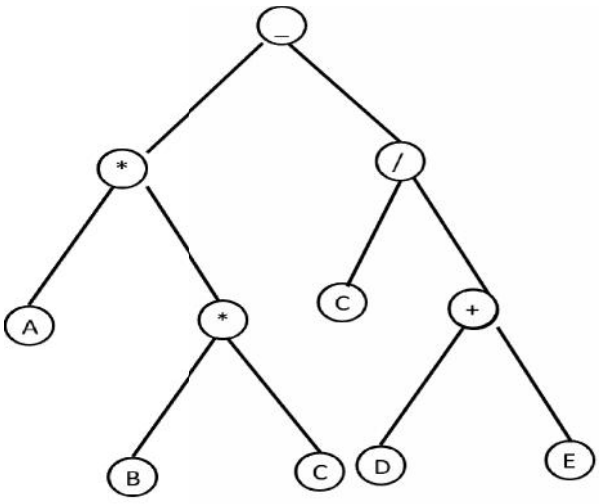
1. A B C * * C D E + / –

2. A (B * C) * C (D + E) / -

3. (A * (B * C)) (C / (D + E)) –

4. (A * (B * C)) – (C / (D + E))

The corresponding binary tree is



Question: Evaluate the postfix form

21 – 342 ÷ + ×

Solution: We have

21 – 342 ÷ + ×

= (2 – 1) 342 ÷ + ×

= 13 (4 ÷ 2) + ×

= 132 + ×

= 1 (3 + 2) ×

= 15 ×

= 1 × 5

= 5

14.4 Traversal Algorithms

Procedures for systematically visiting every vertex of an ordered rooted tree are called traversal algorithms. We have described three of the most commonly used such algorithms, preorder traversal, inorder traversal, and post-order traversal. Each of these algorithms can be defined recursively. We again first define preorder traversal.

Definition 1 Let T be an ordered rooted tree with root r. If T consists only of r, then r is the preorder traversal of T. Otherwise, suppose that T_1, T_2..., T_n are the sub-trees at r from left to right in T. The preorder traversal begins by visiting r. It continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until Tn is traversed in preorder.

The reader should verify that the preorder traversal of an ordered rooted tree gives the same ordering of the vertices as the ordering obtained using a universal address system. Figure 2 indicates how a preorder traversal is carried out. Example 2 illustrates preorder traversal.

2: In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure 3?

Solution: The steps of the preorder traversal of T are shown in Figure 4. We traverse T in preorder by first listing the root a, followed by the preorder list of the sub tree with root b, the preorder list of the sub tree with root c (which is just c) and the preorder list of the sub tree with root d.

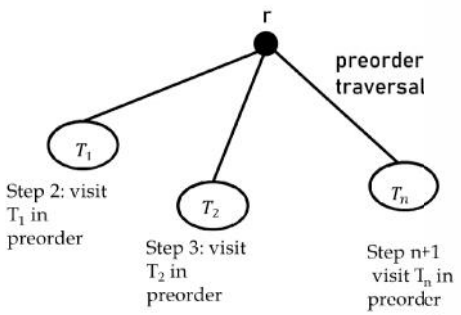


Figure2: Preorder Traversal.

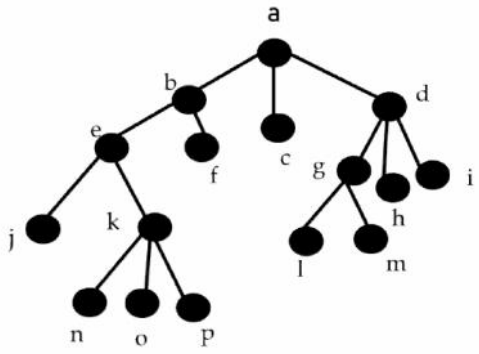


Figure3: The Ordered Rooted Tree T.

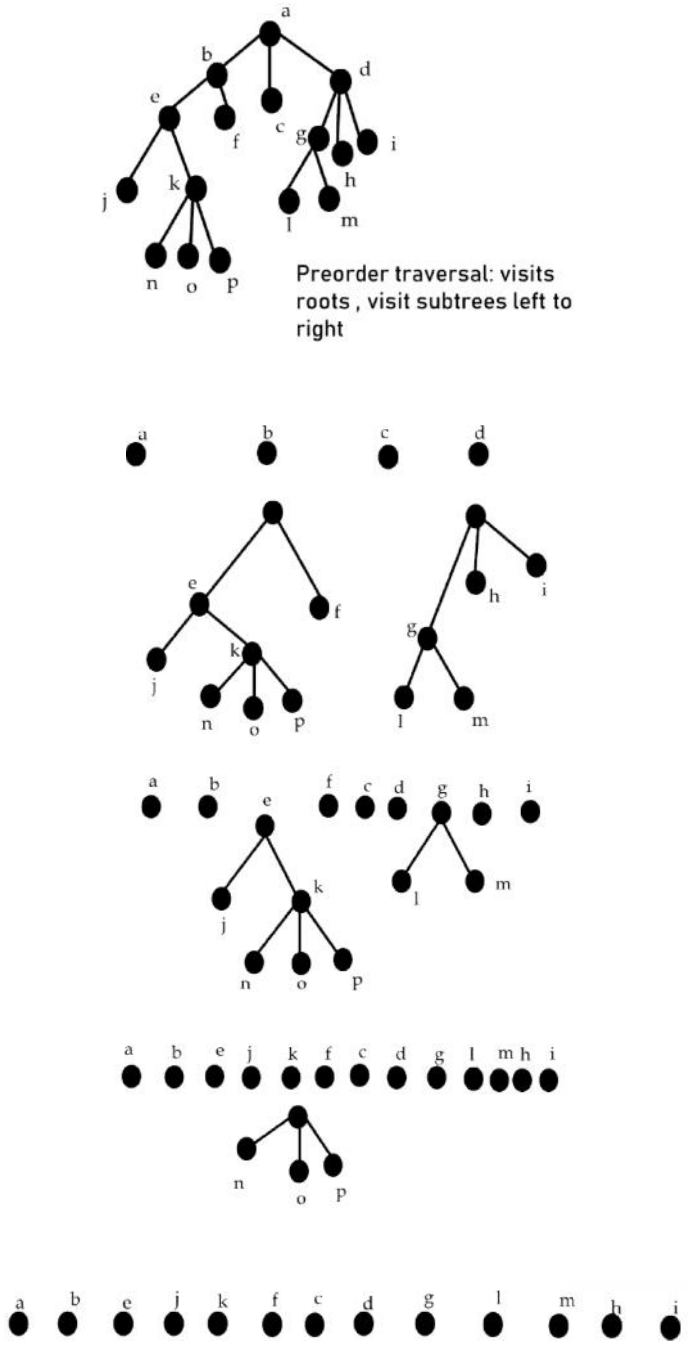


FIGURE 4 The Preorder Traversal of T.

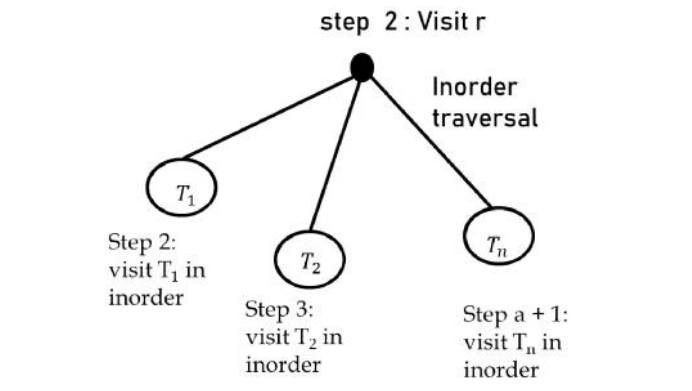


Figure5:Inorder Traversal.

The preorder list of the subtree with root b begins by listing b, then the vertices of the subtree with root e in preorder, and then the subtree with root f in preorder (which is just f). The preorder list of the subtree with root d begins by listing d, followed by the preorder list of the subtree with root g, followed by the subtree with root h (which is just h), followed by the subtree with root i (which is just i). The preorder list of the subtree with root e begins by listing e, followed by the preorder listing of the subtree with root j (which is just j), followed by the preorder listing of the subtree with root k. The preorder listing of the subtree with root g is g followed by l, followed by m. The preorder listing of the subtree with root k is k, n, o, p. Consequently, the preorder traversal of T is a, b, e, j, k, n, o, p, f, c, d, g, l, m, h, i. ▲

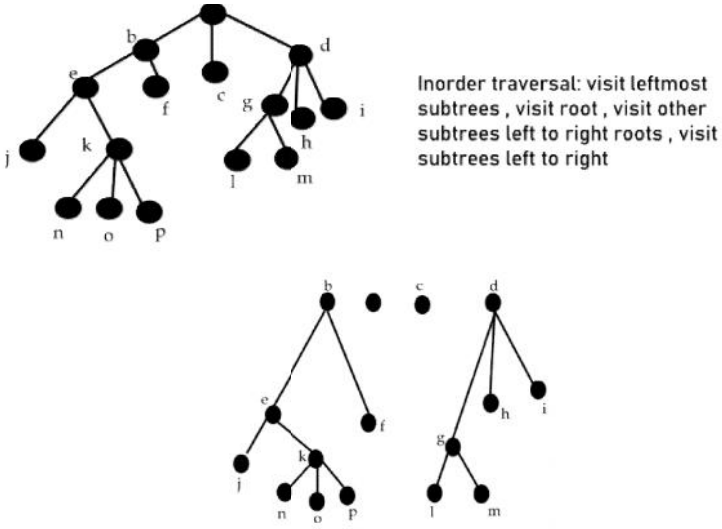
We will now define inorder traversal.

Definition 2 Let T be an ordered rooted tree with root r. If T consists only of r, then r is the inorder traversal of T. Otherwise, suppose that T_1, T_2..., Tn are the subtrees at r from left to right. The inorder traversal begins by traversing T_1 in inorder, then visiting r. It continues by traversing T_2 in inorder, then T_3 in inorder, and finally T_n in inorder.

Figure 5 indicates how inorder traversal is carried out. Example 3 illustrates how inorder traversal is carried out for a particular tree.

3: In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?

Solution: The steps of the inorder traversal of the ordered rooted tree T are shown in Figure 6. The inorder traversal begins with an inorder traversal of the subtree with root b, the root a, the inorder listing of the subtree with root c, which is just c, and the inorder listing of the subtree with root d. The inorder listing of the subtree with root b begins with the inorder listing of the subtree with root e, the root b, and f. The inorder listing of the subtree with root d begins with the inorder listing of the subtree with root g, followed by the root d, followed by h, followed by i. The inorder listing of the subtree with root e is j, followed by the root e, followed by the inorder listing of the subtree with root k. The inorder listing of the subtree with root g is l, g, m. The inorder listing of the subtree with root k is n, k, o, p. Consequently, the inorder listing of the ordered rooted tree is j, e, n, k, o, p, b, f, a, c, l, g, m, d, h, i.



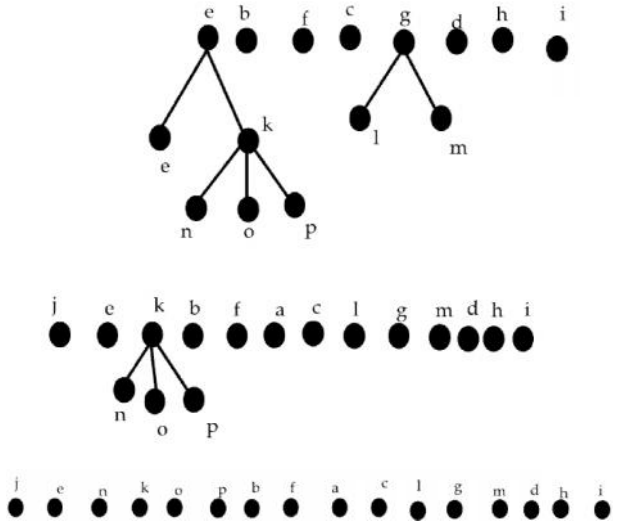


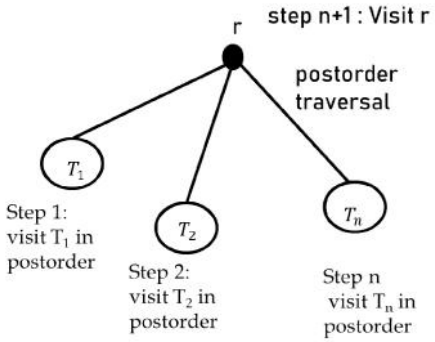
FIGURE 6 The Inorder Traversal of T.

We now define postorder traversal.

Definition 3 Let T be an ordered rooted tree with root r. IfT consists only of r, then r is the postorder traversal of T. Otherwise, suppose that T_1, T_2..., Tn are the subtrees at r from left to right. The postorder traversal begins by traversing T_1 in postorder, then T_2 in postorder, then T_n in postorder, and ends by visiting r.

Figure 7 illustrates how postorder traversal is done.

Example 4 illustrates how postorder traversal works.



Postorder traversal

FIGURE 7 Postorder Traversal.

4: In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown in Figure 3?

Solution: The steps of the postorder traversal of the ordered rooted tree T are shown in Figure 8. Thepostordertraversalbeginswiththepostordertraversalofthesubtreewithrootb, the postorder traversal of the subtree with root c, which is just c, the postorder traversal of the subtree with root d, followed by the root a. The postorder traversal of the subtree with root b begins with the postorder traversal of the subtree with root e, followed by f, followed by the root b. The postorder traversal of the rooted tree with root d begins with the postorder traversal of the subtree with root g, followed by h, followed by i, followed by the root d. The postorder traversal of the subtree with root e begins with j, followed by the postorder traversal of the subtree with root k, followed by the root e. The postorder traversal of the subtree with root g is l, m, g. The postorder traversal of the subtree with root k is n, o, p, k. Therefore, the postorder traversal of T is j, n, o, p, k, e, f, b, c, l, m, g, h, i, d, a. ▲

There are easy ways to list the vertices of an ordered rooted tree in preorder, inorder, and postorder. To do this, first draw a curve around the ordered rooted tree starting at the root, moving along the edges, as shown in the example in Figure 9. We can list the vertices in preorder by listing each vertex the first time this curve passes it. We can list the vertices in inorder by listing a leaf the

first time the curve passes it and listing each internal vertex the second time the curve passes it. We can list the vertices in postorder by listing a vertex the last time it is passed on the way back up to its parent. When this is done in the rooted tree in Figure 9, it follows that the preorder traversal gives a, b, d, h, e, i, j, c, f, g, k, the inorder traversal gives h, d, b, i, e, j, a, f, c, k, g; and the postorder traversal gives h, d, i, j, e, b, f, k, g, c, a. Algorithms for traversing ordered rooted trees in preorder, inorder, or postorder are most easily expressed recursively.

ALGORITHM 1 Preorder Traversal.

procedure preorder (T : ordered rooted tree) r := root of T list r for each child c of r from left to right T (c):= subtree with c as its root preorder(T (c))

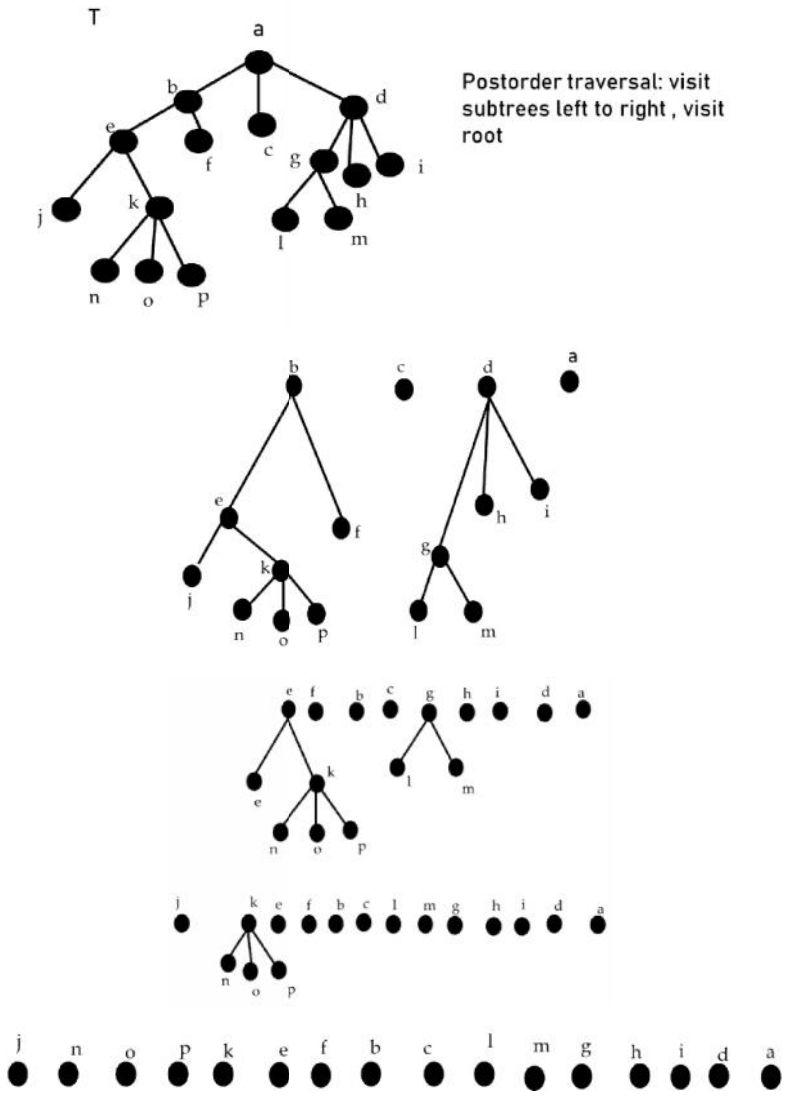


Figure8: The Postorder Traversal of T.

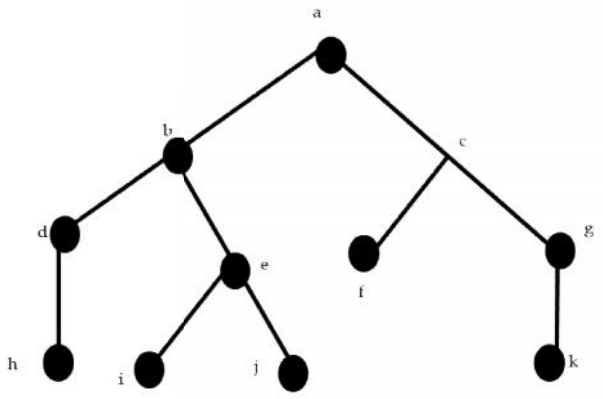


Figure9: A Shortcut for Traversing an Ordered Rooted Tree in Preorder, Inorder, and Postorder.

ALGORITHM 2 Inorder Traversal.

procedure inorder(T : ordered rooted tree) r := root of T if r is a leaf then list r else l := first child of r from left to right T (l):= subtree with l as its root inorder(T (l)) list r for each child c of r except for l from left to right T (c):= subtree with c as its root inorder(T (c))

ALGORITHM 3 Postorder Traversal.

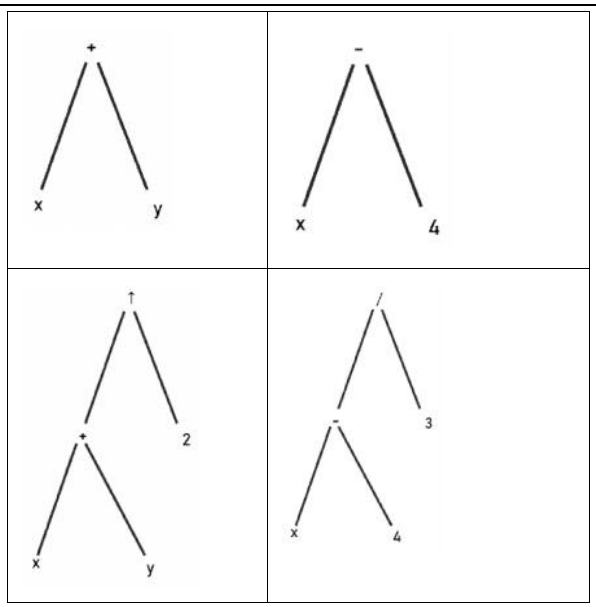
procedure postorder (T: ordered rooted tree) r: = root of T for each child c of r from left to right T (c): = subtree with c as its root postorder (T (c)) list r

Note that both the preorder traversal and the postorder traversal encode the structure of an ordered rooted tree when the number of children of each vertex is specified. That is, an ordered rooted tree is uniquely determined when we specify a list of vertices generated by a preorder traversal or by a postorder traversal of the tree, together with the number of children of each vertex (see Exercises 26 and 27). In particular, both a preorder traversal and a postorder traversal encode the structure of a full ordered many tree. However, when the number of children of vertices is not specified, neither a preorder traversal nor a postorder traversal encodes the structure of an ordered rooted tree (see Exercises 28 and 29).

14.5 Infix, Prefix, and Postfix Notation

We can represent complicated expressions, such as compound propositions, combinations of sets, and arithmetic expressions using ordered rooted trees. For instance, consider the representation of an arithmetic expression involving the operators + (addition), − (subtraction), ∗ (multiplication), / (division), and ↑ (exponentiation). We will use parentheses to indicate the order of the operations. An ordered rooted tree can be used to represent such expressions, where the internal vertices represent operations, and the leaves represent the variables or numbers. Each operation operates on its left and right subtrees (in that order).

5: What is the ordered rooted tree that represents the expression $((x + y) \uparrow 2) + ((x - 4)/3)$? Solution: The binary tree for this expression can be built from the bottom up. First, a subtree for the expression $x + y$ is constructed. Then this is incorporated as part of the larger subtree representing $(x + y) \uparrow 2$. Also, a subtree for $x - 4$ is constructed, and then this is incorporated into a subtree representing $(x - 4)/3$. Finally, the subtrees representing $(x + y) \uparrow 2$



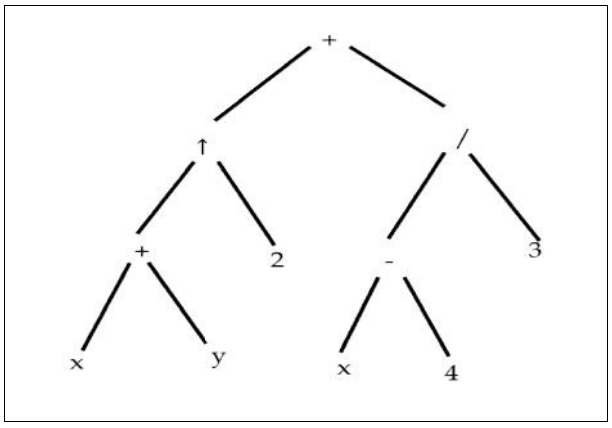


Figure 10 A Binary Tree Representing ((x +y) ↑ 2) +((x −4)/3). and (x −4)/3 are combined to form the ordered rooted tree representing ((x +y) ↑ 2) + ((x −4)/3). These steps are shown in Figure 10. ▲

An inorder traversal of the binary tree representing an expression produces the original expressionwiththeelementsandoperationsinthesameorderastheyoriginallyoccurred, except for unary operations, which instead immediately follow their operands. For instance, inorder traversals of the binary trees in Figure 11, which represent the expressions (x +y)/ (x +3), (x +(y/x)) +3, and x +(y/ (x +3)), all lead to the infix expression x +y/x +3. To make such expressions unambiguous it is necessary to include parentheses in the inorder traversal whenever we encounter an operation. The fully parenthesized expression obtained in this way is said to be in infix form. We obtain the prefix form of an expression when we traverse its rooted tree in preorder. Expressions written in prefix form are said to be in Polish notation, which is named after the Polish logician Jan Lukasiewicz . An expression in prefix notation (where each operation has a specified number of operands), is unambiguous, so no parentheses are needed in such an expression.

6: What is the prefix form for ((x +y) ↑ 2) +((x −4)/3)?

Solution: Weobaintheprefixformforthisexpressionbytraversingthebinarytreethatrepresents it in preorder, shown in Figure 10. These produces+↑+xy2/−x 4 3. ▲ In the prefix form of an expression, a binary operator, such as+, precedes its two operands. Hence, we can evaluate an expression in prefix form by working from right to left. When we encounter an operator, we perform the corresponding operation with the two operands

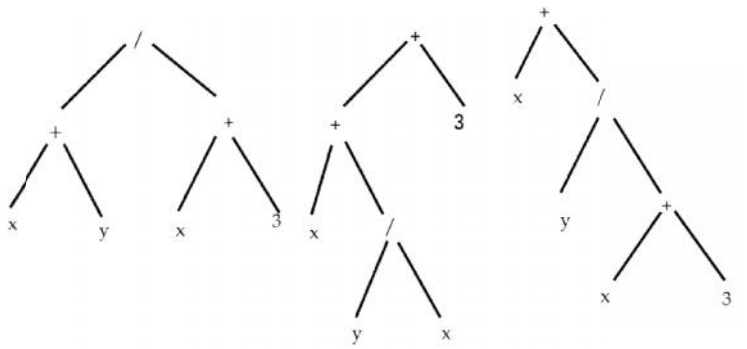


Figure11: Rooted Trees Representing (x +y)/(x +3), (x +(y/x))+3, and x +(y/(x +3)).

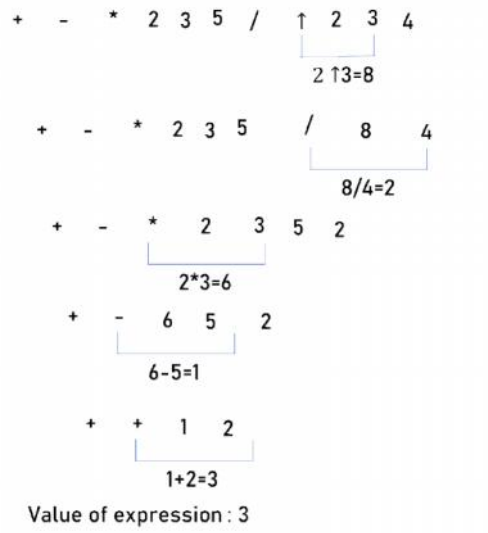


figure 12: Evaluation a prefix expression

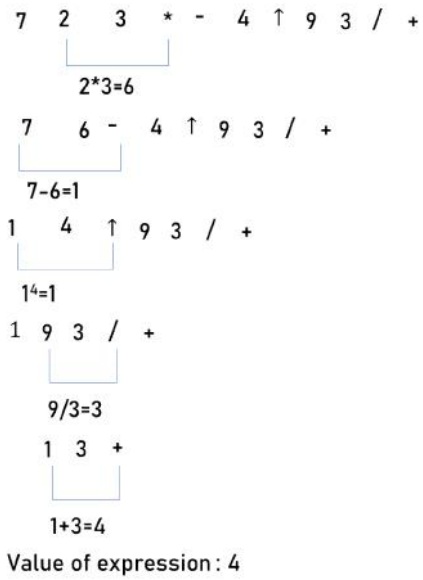


Figure 13 : evaluating a postfix expression

immediately to the right of this operand. Also, whenever an operation is performed, we consider the result a new operand.

7: What is the value of the prefix expression+−∗235 /↑ 2 3 4?

Solution: The steps used to evaluate this expression by working right to left, and performing operations using the operands on the right, are shown in Figure 12. The value of this expression is 3. ▲

We obtain the postfix form of an expression by traversing its binary tree in postorder.

Reverse polish notation was first proposed in 1954 by Burks, Warren, and Wright.

Expressions written in postfix form are said to be in reverse Polish notation. Expressions in reverse Polish notation are unambiguous, so parentheses are not needed. Reverse polish notation was extensively used in electronic calculators in the 1970s and 1980s.

8: What is the postfix form of the expression ((x +y) ↑ 2) +((x −4)/3)?

Solution: The postfix form of the expression is obtained by carrying out a postorder traversal of the binary tree for this expression, shown in Figure 10. This produces the postfix expression: x y +2 ↑ x 4−3/+. ▲

In the postfix form of an expression, a binary operator follows its two operands. So, to evaluate an expression from its postfix form, work from left to right, carrying out operations whenever an operator follows two operands. After an operation is carried out, the result of this operation becomes a new operand.

9: What is the value of the postfix expression 723∗−4 ↑ 93 /+?

Solution: The steps used to evaluate this expression by starting at the left and carrying out operations when two operands are followed by an operator are shown in Figure 13. The value of this expression is 4.

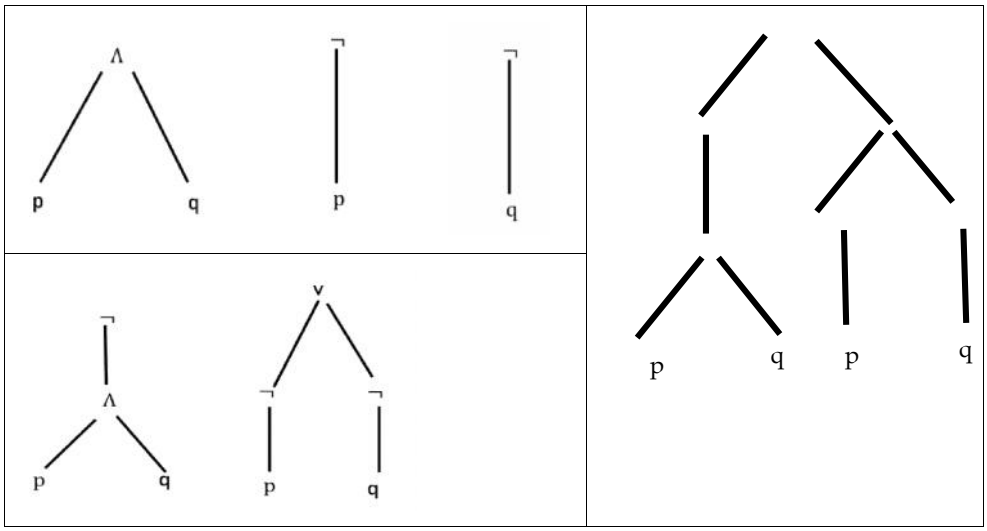


Figure14: Constructing the Rooted Tree for a Compound Proposition.

Rooted trees can be used to represent other types of expressions, such as those representing compound propositions and combinations of sets. In these examples' unary operators, such as the negation of a proposition, occur. To represent such operators and their operands, a vertex representing the operator and a child of this vertex representing the operand are used.

10: Find the ordered rooted tree representing the compound proposition (¬(p∧q)) ↔ (¬p∨¬q). Then use this rooted tree to find the prefix, postfix, and infix forms of this expression.

Solution: Therootedtreeforthiscompoundpropositionisconstructedfromthebottomup.First, subtrees for ¬p and ¬q are formed (where ¬ is considered a unary operator). Also, a subtree for p ∧ q is formed. Then subtrees for ¬(p∧q) and (¬p) ∨(¬q) are constructed. Finally, these two subtrees are used to form the final rooted tree. The steps of this procedure are shown in Figure 14. The prefix, postfix, and infix forms of this expression are found by traversing this rooted tree in preorder, postorder, and inorder (including parentheses), respectively. These traversals give ↔¬∧pq∨¬p¬q, pq∧¬p¬ q¬∨↔, and (¬(p∧q)) ↔ ((¬p) ∨(¬q)), respectively. ▲

Because prefix and postfix expressions are unambiguous and because they can be evaluated easily without scanning back and forth, they are used extensively in computer science. Such expressions are especially useful in the construction of compilers.

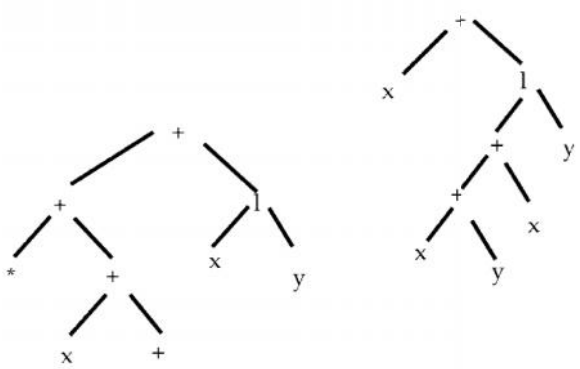
Summary

- We have learned what is tree traversal.
- We have learnedarethreetraversal which are called preorder, inorder, and postorder.
- We have learned what areInfix, Prefix, and Postfix Notation

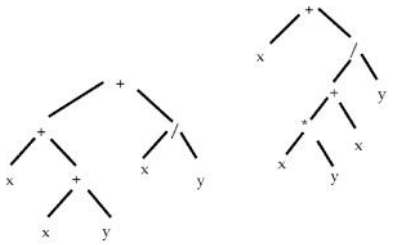
Self Assessment

1. Represent the expressions (x +xy) +(x/y) and x + ((xy +x)/y) using binary trees.

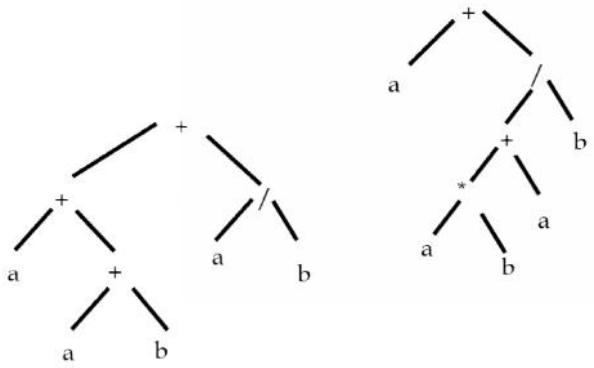
A.



B.



C.



D. None of these

2. Represent the expressions b) prefix notation.

A. ++x∗xy/xy, +x/+∗xyxy

B. ++x∗xy/xy, +x/+∗xyxy

C. ++a∗by/ny, +z/+∗xyz

D. ++x∗xyz/xyz, +x/+∗xy

3. Represent the expressions c) postfix notation.

A. xxy∗+xy/+, xxy∗x+y/+

B. yzy∗+zy/+, xy∗x+z/+

C. abc∗+ab/+, abc∗a+z/+

D. All of these

4. Represent the expressions d) infix notation.

A. ((m + (m ∗n)) +(m/n)), (m + (((m ∗n) +x)/y))

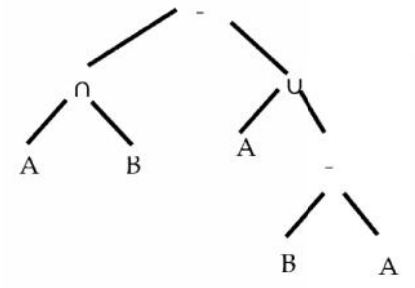
B. ((x + (a ∗b)) +(a/b)), (x + (((a ∗b) +a)/b))

C. ((q + (q ∗r)) +(q/r)), (x + (((q ∗r) +x)/y))

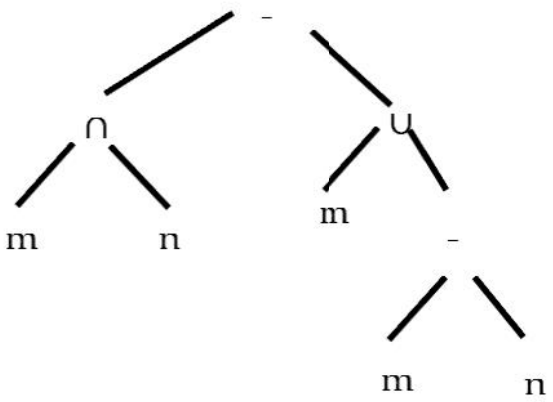
D. ((x + (x ∗y)) +(x/y)), (x + (((x ∗y) +x)/y))

5. a) Represent $(A \cap B) - (A \cup (B - A))$ using an ordered rooted tree.

A.

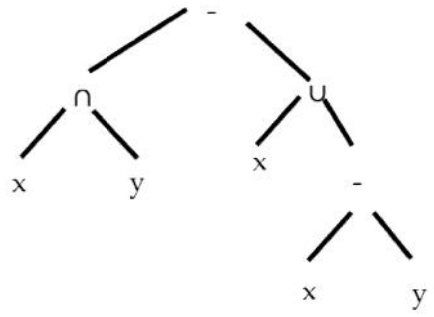


B.



.

C.



D. All of above

6. Write this expression $(A \cap B) - (A \cup (B - A))$ in prefix notation

A. −∩AB∪A−BA.

B. −∩B∪A−Bc

C. −∩ABCD−BA

D. −∩ABCD−DA

7. Write this expression $(A \cap B) - (A \cup (B - A))$ in postfix notation.

A. AB∩ABA−∪−

B. BC∩BA−∪−

C. AD ∩ AD A - ∪ –

D. Both C and B

8. Write this expression $(A \cap B) - (A \cup (B - A))$ in infix notation.

A. ((D∩C) −(C∪(D−A)))

B. ((A∩B) −(A∪(B−A)))

C. ((X∩Y) −(X∪(X−A)))

D. None of these

9. In how many ways can the string A∩B −A∩B −A be fully parenthesized to yield an infix expression?

A.16

B. 18

C. 14

D. Both b and c

10. What is the value of the prefix expression −∗2/843 ?

A. 1

B. 3

C. 0

D. 9

11. What is the value of the prefix expression↑−∗33∗425?

A. 10

B. 20

C. 89

D. 1

12. What is the value of the prefix expression +−↑32↑23/6−42?

A. 1

B. 2

C. 8

D. 4

13. What is the value of the prefix expression ∗+3+3 ↑ 3+333?

A. 2212

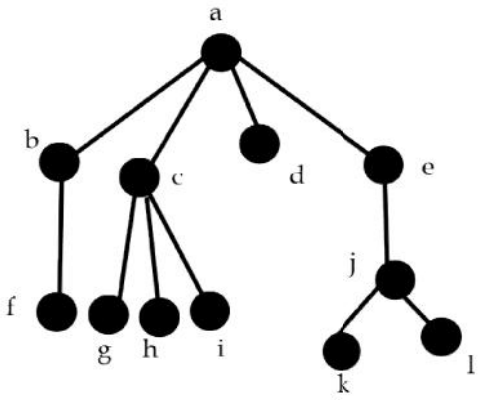
B. 4532

C. 2205

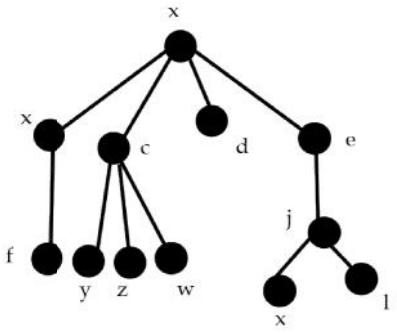
D. 1001

14. Construct the ordered rooted tree whose preorder traversal is a, b, f, c, g, h, i, d, e, j, k, l, where a has four children, c has three children, j has two children, b and e have one child each, and all other vertices are leaves.

A.



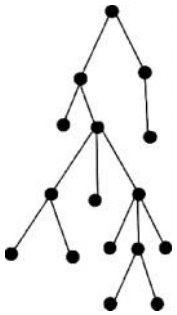
B.



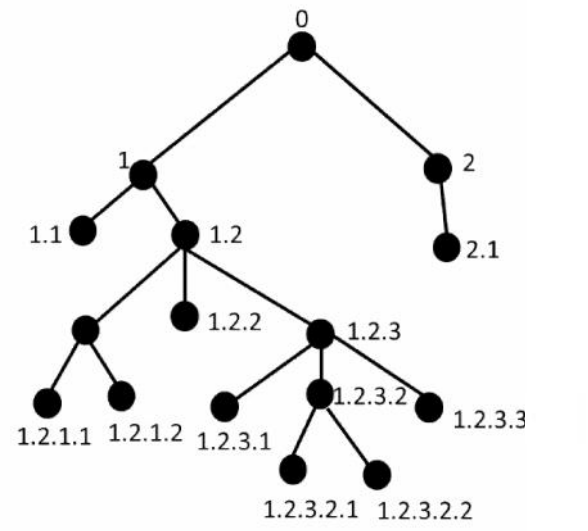
C. both a and b

d) none of these

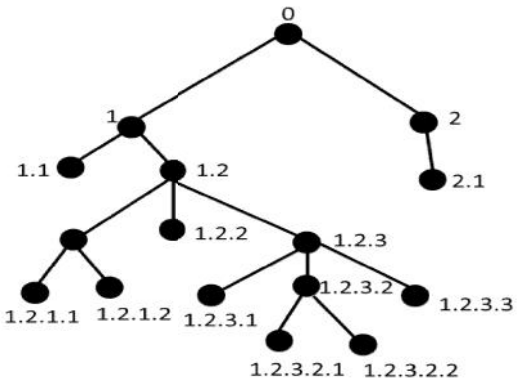
15. Constructedthe universal address system for the given ordered rooted tree and use this to order its vertices using the lexicographic order of their labels



A. The order is 0 < 1 < 1.1 < 1.2 < 1.2.1 < 1.2.1.1 < 1.2.1.2 < 1.2.2 < 1.2.3 < 1.2.3.1 < 1.2.3.2 < 1.2.3.2.1 < 1.2.3.2.2 < 1.2.3.3 < 2 < 2.1.

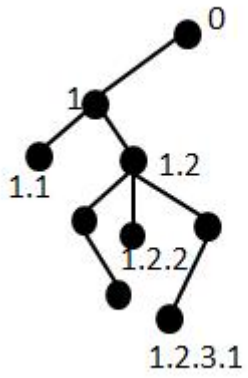


B. The order is 0 < 1 < 1.1 < 1.2 < 1.2.1 < 1.2.1.1 < 1.2.1.2 < 1.2.2 < 1.2.3 < 1.2.3.1 < 1.2.3.2 < 1.2.3.2.1 < 1.2.3.2.2 < 1.2.3.3.



C. See the comments for the solution to Exercise 1. The order is 0 < 1 < 1.1 < 1.2 < 1.2.1 < 1.2.1.1 < 1.2.1.2 < 1.2.2 < 1.2.3.

D.



Answers for Self Assessment

1.	B	2.	B	3.	A	4.	D	5.	A
6.	A	7.	A	8.	B	9.	C	10.	A
11.	D	12.	D	13.	C	14.	A	15.	A

Review Questions

Q1**.** Suppose that the address of the vertex v in the orderedrooted tree T is 3.4.5.2.4.a) At what level is v?b) What is the address of the parent of v?c) What is the least number of siblings v can have?d) What is the smallest possible number of vertices in Tif v has this address?e) Find the other addresses that must occur.

.

Q2. Can the leaves of an ordered rooted tree have the followinglist of universal addresses? If so, construct such anordered rooted tree.a) 1.1.1, 1.1.2, 1.2, 2.1.1.1, 2.1.2, 2.1.3, 2.2, 3.1.1,3.1.2.1, 3.1.2.2, 3.2

Q3. a) Represent the expression $((x+2) \uparrow 3) * (y - (3+x)) - 5$ using a binary tree.Write this expression in
b) prefix notation.
c) postfix notation.
d) infix notation.

This implies, $g(T)\alpha = \ < \alpha >$

Also, taking $g(x) = x$, we have, $g(T)\alpha = c'\alpha$

That is, $T\alpha = c'\alpha$

So that, α is characteristic vector of T.

Conversely, let α is characteristic vector of T.

Then there exist some $c \in F$ such that $T\alpha = c\alpha$

In that case, $g(T)\alpha = g(c)\alpha$

This implies, $g(T)\alpha = \ < \alpha >$ for all $g \in F[x]$

That is, $Z(\alpha; T) =< \alpha >$

Proof of part (iii)

For $T = I$,

$Z(\alpha; I) = \{g(I)\alpha; g \in F[x]\}$

Let $g(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$

Then

$$g(I) = a_0I + a_1I^2 + \cdots + a_nI^n$$
$$= a_0 + a_1 + \cdots a_n)I = cI$$

So that,

$$Z(\alpha; I) = \{cI\alpha; c \in F\}$$
$$= \{c\alpha; c \in F\}$$
$$=< \alpha >$$

Since $\alpha \neq 0$,

$\{\alpha\}$ is linearly independent.

So, $Z(\alpha; T)$ has basis $\{\alpha\}$

That is, $Z(\alpha; T)$ is $1-$dimensional.

Further, if dimension $V > 1$, $T = I$

then $Z(\alpha; T)$ is one dimensional

but since dimension $V > 1$.

That is, $Z(\alpha; T)$ is a proper subspace of V.

So, $V \neq Z(\alpha; T)$ for any α,

hence, identity operator has no cyclic vector if dim $V > 1$

Example 14.1.4: An operator which has a cyclic vector.

Let T be an operator on F^2 which is represented in the standard ordered basis by matrix

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Proof: Let $\alpha = (1, 0)$

Claim: $Z(\alpha; T) = F^2$

$Z(\alpha; T) = \{g(T)\alpha | g \in F[x]\}$ is a subspace of F^2.

Therefore, $Z(\alpha; T) \subseteq F^2$ ... (1)

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-521360
Fax.: +91-1824-506111
Email: odl@lpu.co.in