M.Sc. Mathematics

MAL-637

ADVANCED DISCRETE MATHEMATICS

Directorate of Distance Education

Guru Jambheshwar University of Science & Technology.

HISAR-125001

CONTENTS

Author: Dr. Vizender Singh
(Assitant Professor and Programme Coordinator M.Sc. Mathematics)
Directorate of Distance Education
Guru Jambheshwar University of Science & Technology
Hisar, Haryana-125001.

Vetter: Dr. Renu Muwal (Assistant Professor)
Department of Mathematics,
Guru Jambheshwar University of Science & Technology
Hisar, Haryana-125001.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 1 **Written by Dr. Vizender Singh**

MATHEMATICAL LOGICS

Structure:

1.0 Learning Objectives

In this chapter, various basic aspects of logic will be discussed. The reader will learn about Logical statements, their symbolic representations, construction of truth tables, laws of algebra of Propositions, Quantifiers etc. Examples are also given to illustrate these topics.

1.1 Introduction

Logic is the study of principles and techniques of reasoning. It was Aristotle, the Greek scientist and philosopher who first time studied the logical reasoning. The aim of logic is to provide rule by which one can judge whether a particular reasoning is valid or not. It is also important starting point of study of discrete mathematics. Logics are basis of many areas of computer science such as digital logic design, automata theory and computability, artificial intelligence etc. One of the components of logic is proposition calculus which deals with statement with value true or false and is concerned with proposition analysis.

1.2 Some Basic Definitions

1.2.1 Proposition or Statement

Definition: A proposition or statement is a declarative sentence that is either true or false, but not both. For instance, "Two plus two equals four" and "Two plus two equals five" are both statements, because first is true and second is false. But " $x + y < 7$" is not a statement, because it is true for some values of x and y, but false for other values. The truth and falsity of statement are called its **truth value**. Since only two truth values are admitted, this logic sometimes is also called **two valued logic**. It is to be noted here that, questions, exclamations and commands are not propositions. In support consider the following examples:

(a) The sun rises in the west.

(b) $\{2, 3, 5\} \subset \{ 3, 5, 8, 9\}$

(c) Do you speak English?

(d) $4 – x = 7$

(e) Close the door

(f) What a hot day?

(g) We shall have chicken for dinner.

The equality (a) and inclusion (b) are propositions, as first is true and second is false. But (c) is a question, not a declarative sentence; hence it is not a statement.

The equality (d) is declarative, but not proposition as its truth value depends upon value of variable x.

The sentence (e) is not a statement, it is command.

The sentence (f) is not a statement, it is exclamation.

The sentence (g) is a statement as it is neither true of false but not both, but one has to wait till dinner to know it is true or false.

The simple propositions can be represented by letters p, q, r, …….. known as its propositional variables. The propositional variables can assume only two values: 'true' denoted by T or 1 and 'false' denoted by F or 0. If the letter p denotes proposition "The sun sets in the west", then instead of saying that the proposition is true, one can simply say that p has value F.

1.2.2 Atomic and Compound Propositions

Definition: A proposition consisting of single propositional variable or constant is called atomic (primary, primitive) proposition or simply proposition, i.e., cannot be further subdivided. A proposition obtained from the combinations of two or more propositions by means of logical operations or connectives of two or more propositions or by neglecting a single proposition is referred to compound (composite, molecular) proposition.

1.2.3 Connectives

Definition: The word and phrases (symbols) used to form compound propositions are called connective. There are five basic connectives called Conjunction, Disjunction, Negation, conditional and Bi-conditional. The following table gives the description:

Symbol Used	Connective Word	Nature of compound statement formed by the connective	Symbolic Form	Negation
~, ¬, ⅂, N	not	Negation	~p	~ (~p) = p
∧	and	Conjunction	p ∧ q	(~p) ∨ (~q)
∨	or	Disjunction	p ∨ q	(~p) ∧ (~q)
→	If ….. then	Conditional	p → q	(p) ∧ (~q)
↔	If and only if	Bi-conditional	p ↔ q	[(p) ∧ (~q)] ∨ [(q) ∧ (~p)]

1.3 Truth table

A table that shows truth value of compound proposition for all possible cases is called **truth table**.

The columns of the table are for variables p, q, r, …. and the number of rows depends upon number of variables. For 2 variables, 4 rows are necessary; for 3 variables, 8 rows are necessary; and in general for n variables, 2^n rows are required. There is a column for each elementary stage of construction of proposition. The truth value at each stage is determined from previous stage of definition of connectives. The truth value of proposition appears in last column.

1.4 Basic Logical Operations

(a) Negation (b) Conjunction (c) Disjunction

1.4.1 Negation

Definition: If p is any proposition, then the negation of p, denoted by ~ p and is read as not p, is the proposition which is false when p is true and true when p is false.

Truth Table for Negation:

p	~ p
T	F
F	T

It is to be noted here that the word "not" is not a connective, since it does not join two statements and ~ p is not really a compound statement. However, not is a unary operation for collection of statements and ~ p is a statement if p is considered a statement.

For instance, consider the statement:

p : Delhi is in India.

The negation of p is the statement

~p : It is not the case that Delhi is in India.

Normally it is written as

~p : Delhi is not in India.

Example 1.1 Write the negation of the following statements:

(a) p: 3 + 5 > 1

(b) q: It is hot.

Solution. (a) ~ p: 3 + 5 ≤ 1

(b) ~ q: It is not hot.

1.4.2 Conjunction

Definition: If p and q are two propositions, then conjunction of p and q is the compound proposition denoted by p ∧ q and read as "p and q". The compound proposition is true if p and q are both true, otherwise it is false even if one of component is false.

Truth Table for Conjunction:

p	q	p ∧ q
T	T	T
F	T	F
T	F	F
F	F	F

Example 1.2 Let

p: This child is boy.

q: This child is intelligent .

be two statements.

Then conjunction is p ∧ q: This child is boy and intelligent.

Example 1.3 If

p: Delhi is capital of China.

q: A decade is of 10 years.

Then p is false and q is true.

The conjunction is false and written as p ∧ q: Delhi is capital of India and a decade is of 10 years.

Example 1.4 Consider the following statements:

(i) The Red fort is in Delhi and 3 is prime number.

(ii) Paris is in France and $\sqrt{2}$ is rational number.

(iii) Delhi is in China and 2 + 2 = 4

(iv) Delhi is in England and 2 + 2 = 5.

Solution. The above four statements are combined by connective "and". Therefore the compound statement shall be conjunction. It is obvious that only statement (i) is true and rest of all (ii) - (iv) are false. The truth table is

p	q	p ∨ q
T	T	T
T	F	F
F	T	F
F	F	F

1.4.3 Disjunction

Definition: If p and q are two propositions, then conjunction of p and q is the compound proposition denoted by p ∨ q and is read as "p or q". The compound proposition is false when both of p and q are false; otherwise it is true even if one of component is true.

Truth Table for disjunction:

p	q	p ∨ q
T	T	T
F	T	T
T	F	T
F	F	F

Example 1.5 If

p: Delhi is capital of China.

q: A decade is of 10 years

Then p is false and q is true.

The disjunction is true and written as p ∨ q : Delhi is capital of India or a decade is of 10 years.

It is clear from above example that totally unrelated statements can be joined with the help of connective "or".

1.5 Translation from Language to Symbols

Example 1.7 Write each of the following statement in symbolic form:

(a) It is not hot but it is sunny

(b) It is neither hot nor sunny.

Solution. (a) It is to be noted by reader that in logic the word "but" and "and" has same meaning. In general the word "but" is used in place of word "and" when the part of sentence that follows is in some way unexpected.

The given sentence is equivalent to p: It is not hot and it is sunny, which is written as $\sim p \wedge q$.

(b) The phrase neither A nor B means the same as not A and not B. Symbolically the given statement is written as $\sim p \wedge \sim q$.

1.6 Statement form

Definition: A "Statement form" or "Propositional form" is an expression made up of **statement variables** (such as p, q, r, ….) and logical connectives (such as ~, $\wedge$, $\vee$) that becomes a statement when actual statements are substituted for the component statement variable. The **truth table** for a given statement form displays the truth values that correspond to the different combinations of truth values for the variables.

Example 1.8 Construct the truth table for statement form

$$(p \wedge q) \vee \sim r$$

Solution. The truth table is

p	q	r	$p \wedge q$	$\sim r$	$(p \wedge q) \vee \sim r$
T	T	T	T	F	T
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	F	T	T
F	T	T	F	F	F
F	T	F	F	T	T
F	F	T	F	F	F
F	F	F	F	T	T

Example 1.9 Construct the truth table for statement form

$$p \wedge (\sim q \vee q)$$

Solution. The truth table is

p	q	~q	~q ∨ q	p ∧ (~q ∨ q)
T	T	F	T	T
T	F	T	T	T
F	T	F	T	F
F	F	T	T	F

Example 1.10 Construct the truth table for statement form

$$\sim (p \vee q) \ \vee (\sim p \wedge \sim q)$$

Solution. The truth table is

p	q	~p	~q	p ∨ q	~(p ∨ q)	(~p ∧ ~q)	~(p ∨ q) ∨ (~p ∧ ~q)
T	T	F	F	T	F	F	F
T	F	F	T	T	F	F	F
F	T	T	F	T	F	F	F
F	F	T	T	F	T	T	T

1.7 Logical Equivalence of Propositions

Definition: If two propositions P (p, q, ……) and Q (p, q, …....) , where p, q, ….. are propositional variables, have same truth value for all possible cases, then the propositions P and Q are called logically equivalent or simply equivalent and denoted by

P (p, q, ……) ≡ Q (p, q, …....)

Or simply we say that two compound propositions are said to be logically equivalent if they have identical truth table.

It is to be noted that two equivalent propositions can be replaced by one another.

Example 1.11 Show that the statements ~ p ∧ ~ q and ~ (p ∧ q) are not logically equivalent.

Solution. The truth table for both forms is:

p	q	~p	~q	p ∧ q	~(p ∧ q)	~p ∧ ~q
T	T	F	F	T	F	F
T	F	F	T	F	T	F
F	T	T	F	F	T	F
F	F	T	T	F	T	T

The propositions are not equivalent as they have different truth value in row 2 and row 3.

If one consider ~ p ∨ ~ q, the column in truth table is

~ p ∨ ~ q
F
T
T
T

Showing that ~ (p ∧ q) ≡ ~p ∨ ~q i.e., ~ (p ∧ q) and ~p ∨ ~q are logically equivalent. Also it is obvious that ~ (p ∨ q) ≡ ~p ∧ ~q. Both of these logical equivalence are known as **De Morgan's Laws of logic**.

Example 1.12 Use **De Morgan's Law of logic** to write the negation of inequality $-4 < x \leq 2$.

Solution. The given statement is equivalent to $-4 < x$ and $x \leq 2$.

Using De Morgan's Law of logic, the negation is $-4 \not< x$ or $x \not\leq 2$, which is equivalent to $-4 \geq x$ or $x > 2$.

Example 1.13 Use **De Morgan's Law of logic** to write the negation of statement

p: Veronica is a girl and Veronica is beautiful.

Solution. Negation of statement p is

~p: Veronica is not a girl or Veronica is not beautiful.

It is to be noted that it will be wrong to write ~p: Veronica is not a girl and beautiful as the connective "and" and "or" are used between complete statements.

1.8 Algebra of propositions

Propositions satisfy various laws as in following table. A compound proposition can be replaced by one that is equivalent to it without changing its truth value. These laws are helpful in simplifying expressions. All the laws except involution law occur in pair called dual pairs. The dual of each expression can be found by replacing all "T" by "F" and all "F" by "T" and replacing all "∧" by "∨" and all "∨" by "∧". All of these laws can easily be established by using truth table.

Sr. No.	Name of Law	Primal Form	Dual Form
1.	Idempotent Law	$p \vee p = p$	$p \wedge p = p$
2.	Identity Law	$p \vee F = p$	$p \wedge T = p$
3.	Dominant Law	$p \vee T = T$	$p \wedge F = F$
4.	Complement Law	$p \vee \sim p = T$	$p \wedge \sim p = F$
5.	Associative Law	$(p \vee q) \vee r = p \vee (q \vee r)$	$(p \wedge q) \wedge r = p \wedge (q \wedge r)$
6.	Distributive Law	$p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$	$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$
7.	Absorption Law	$p \vee (p \wedge q) = p$	$p \wedge (p \vee q) = p$
8.	Commutative Law	$p \vee q = q \vee p$	$p \wedge q = q \wedge p$
9.	De Morgan Law	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
10.	Involution Law	$\sim(\sim p) = p$	

Example 1.14 Prove Distributive law using truth table

$$p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$$

Solution. The truth table is

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$(p \vee q)$	$(p \vee r)$	$(p \vee q) \wedge (p \vee r)$
T	T	T	T	T	T	T	T

T	T	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	T	T	T	T	T
F	T	F	F	F	T	F	F
F	F	T	F	F	F	T	F
F	F	F	F	F	F	F	F

It is clear from truth table that entries in 5^{th} and last column are same, which proves the establishment of Distributive law.

Example 1.15 Prove absorption law using truth table

$$p \vee (p \wedge q) = p \text{ and } p \wedge (p \vee q) = p$$

Solution. The truth table is

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

The truth table for p and $p \vee (p \wedge q)$ is same, so $p = p \vee (p \wedge q)$.

The second case $p \wedge (p \vee q) = p$ can be established on similar lines.

Example 1.16 Prove commutative law using truth table

$$(p \wedge q) = (q \vee p)$$

Solution. The truth table is

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

F	F	F	F

The truth table for $(p \wedge q)$ and $(q \vee p)$ is same so $(p \wedge q) = (q \vee p)$.

Example 1.17 Simplify the logical statement

$$\sim (\sim p \wedge q) \wedge (p \vee q).$$

Solution. We have

$\sim (\sim p \wedge q) \wedge (p \vee q) = (\sim (\sim p) \vee \sim q)) \wedge (p \vee q)$ (De Morgan's Law)

$= (p \vee \sim q) \wedge (p \vee q)$ (Involution Law)

$= p \vee (\sim q \wedge q)$ (Distributive law)

$= p \vee c$ (Complement Law)

$= p$ (Identity Law)

1.9 Tautology and Contradiction

Definition: A compound proposition P $(p_1, p_2, p_3, \ldots\ldots, p_n)$ where $p_1, p_2, p_3, \ldots\ldots, p_n$ are variables is called tautology, if it is true for every truth assignment for $p_1, p_2, p_3, \ldots\ldots, p_n$. P is called contradiction, if it is false for every truth assignment for $p_1, p_2, p_3, \ldots\ldots, p_n$.

Example 1.18 Consider the statement: $p \vee \sim p$

The truth table for statement is

p	~ p	p ∨~ p
T	F	T
F	T	T

All T's

Hence $p \vee \sim p$ is a tautology.

Example 1.19 Consider the statement: $p \wedge \sim p$

The truth table for statement is

p	~ p	p ∧ ~ p
T	F	F
F	T	F

All F's

Hence $p \vee \sim p$ is a contradiction.

Remark 1.1 If τ and c denote tautology and contradictions, respectively, then it is obvious that $\sim \tau = c$ and $\sim c = \tau$.

Remark 1.2 If τ and c denote tautology and contradictions, then the truth table for $p \wedge \tau$ and $p \wedge c$ are:

p	τ	$p \wedge \tau$
T	T	T
F	T	F

Same truth Value

p	c	$p \wedge c$
T	F	F
F	F	F

Same truth Value

Hence, we conclude, $p \wedge \tau = p$ and $p \wedge c = c$.

Following the similar lines, it is easy to prove, $p \vee \tau = \tau$ and $p \vee c = p$. These four are called universal bound laws or identity laws.

1.10 Conditional Propositions

Definition: If p and q are propositions, then the compound proposition " if p then q", denoted by $p \rightarrow q$ is called **conditional proposition** or **implication**, which is false when p is true and q is false and true otherwise. The proposition p is called **hypothesis** or **antecedent** whereas the proposition q is called **conclusion** or **consequence**.

The truth table for $p \rightarrow q$ is

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

The alternative terminology used to express connective "if……then" are the following:

1. p is sufficient for q. 2. p only if q.
3. q is necessary for p. 4. q if p.
5. q follows for p 6. q is consequence of p.

Example 1.20 Consider the statements:

(i) If tomorrow is Monday then today is Sunday.

(ii) If you work hard then you will get success.

Here

p: Tomorrow is Monday is antecedent.

q: Today is Sunday is consequent.

p: You work hard is antecedent.

q: You will get success is consequent.

Example 1.21 Which of the following propositions are true and which are false?

(a) If the Earth is round then the Earth travels round the Sun.

(b) If Alexander Graham Bell invented telephone, then tigers have wings.

(c) If tigers have wings, then RDX is dangerous.

Solution. (a) True.

Let

p: The earth is round

q: The earth travels round the sun

Here p is true and q is true and hence the conditional proposition is true.

(b) False.

Let

p: Alexander Graham Bell invented telephone.

q: Tigers have wings.

Here p is true and q is false and hence the conditional proposition is false.

(c) True.

Let

p: Tigers have wings.

q: RDX is dangerous.

Here p is false and q is true and hence the conditional proposition is true.

Example 1.22 Construct truth table for

(i) $(p \vee \sim q) \rightarrow p$

(ii) $(\sim(p \wedge q) \vee r) \rightarrow \sim p$

Solution: (i) The truth table of the given following compound statement is shown below:

p	q	~q	p ∨ ~q	(p ∨ ~q) → p
T	T	F	T	T
T	F	T	T	T
F	T	F	F	T
F	F	T	T	F

(ii) The truth table of the given compound statement is shown below:

p	q	r	(p ∧ q)	~ (p ∧ q)	~ (p ∧ q) ∨ r	(~ (p ∧ q) ∨ r) → ~p
T	T	T	T	F	T	F
T	T	F	T	F	F	T
T	F	T	F	T	T	F
T	F	F	F	T	T	F
F	T	T	F	T	T	T
F	T	F	F	T	T	T
F	F	T	F	T	T	T
F	F	F	F	T	T	T

Example 1.23 Use truth table to show that

$$p \rightarrow q \equiv \sim p \vee q$$

Solution. The truth table of $p \rightarrow q$ and $\sim p \vee q$ is shown below

p	q	~ p	p → q	~ p ∨ q
F	F	T	T	T
F	T	T	T	T
T	F	F	F	F

T	T	F	T	T

1.11 Converse, Inverse and Contrapositive of Conditional Propositions

1.12 Biconditional Propositions

Definition: If p and q are propositions, then the compound proposition " p if and only if q", denoted by $p \leftrightarrow q$ is called **biconditional proposition**. The biconditional statement $p \leftrightarrow q$ can also be stated as " p is necessary and sufficient condition for q" which is true when both p and q are true or both p and q are false.

The truth table for bi-conditional statement is as following:

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

Example 1.24 Use truth table to show that

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p).$$

Solution. The truth table of logical equivalence of expression is given by:

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	T
T	F	F	F	T	F
F	T	F	T	F	F
F	F	T	T	T	T

Example 1.25 Prove that $p \leftrightarrow q \equiv (p \wedge q) \rightarrow (p \vee q)$ by using

(a) truth table (b) algebra of propositions.

Solution. (a) The following truth table shows that both the expressions are logically equivalent:

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

(b) $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

$\equiv (\sim p \vee q) \wedge (\sim q \vee p)$

$\equiv [(\sim p \vee q) \wedge \sim q] \vee [(\sim p \vee q) \wedge p]$ (Distributive law)

$\equiv [\sim q \wedge (\sim p \vee q)] \vee [p \wedge (\sim p \vee q)]$ (Commutative law)

$\equiv [(\sim q \wedge q) \vee (\sim q \wedge \sim p)] \vee [(p \wedge \sim p) \vee (p \wedge q)]$ (Distributive law)

$\equiv [F \vee (\sim q \wedge \sim p)] \vee ([F \vee (p \wedge q)]$ (Complement law)

$\equiv (\sim q \wedge \sim p) \vee (p \wedge q)$ (Identity law)

$\equiv [\sim (p \vee q)] \vee (p \wedge q)$ (De Morgan law)

$\equiv (p \wedge q) \rightarrow (p \vee q)$

1.13 Logical Equivalence using Biconditional Propositions

Definition: If two propositions P (p, q, ……) and Q(p, q, …....) are such that $P \leftrightarrow Q$ is a tautology, where p, q, ….. are propositional variables, then the propositions P and Q are called logically equivalent or simply equivalent and denoted by

$P(p, q, \ldots\ldots) \equiv Q(p, q, \ldots....)$ or $P(p, q, \ldots\ldots) \Leftrightarrow Q(p, q, \ldots....)$.

It is to be noted that two equivalent propositions can be replaced by one another.

1.14 Order of Precedence for Logical Connectives

In general, parentheses are used to specify the order in which the logical operators in a compound statement are to be applied. For example, $(p \vee q) \wedge (\sim r)$ is the conjunction of $p \vee q$ and $\sim r$. To avoid the use of excessive number of parentheses, the rule of application of logical operator is given below:

(a) The negation operator has precedence over all logical operators. Thus $\sim p \wedge q$ means $(\sim p) \wedge (q)$, not $\sim (p \wedge q)$.

(b) The conjunction operator has precedence over disjunction operator. Thus p ∧ q ∨ r means (p ∧ q) ∨ r, not p ∧ (q ∨ r).

(c) The conditional and Bi-conditional operators have lower precedence than other operators. Among them → has precedence over ↔.

Thus the hierarchy of operations of logical connectives can be represented as: ~, ∧, ∨, → , ↔

1.15 Negation of Compound Statements

1.15.1 Negation of Conjunction:

The negation of conjunction p ∧ q is the disjunction of negation of p and negation of q. Mathematically, we write

$$\sim (p \wedge q) = \sim p \vee \sim q$$

The truth table of the above negation is

p	q	(p ∧ q)	~(p ∧ q)	~ p	~ q	~ p ∨ ~ q
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Example 1.26 Write the negation of both of following conjunction:

(a) Delhi is in India and London is in England.

(b) 3 + 4 = 7 and 8 < 11.

Solution. (a) Let p : Delhi is in India and q : London is in England.

The conjunction of statement p and q is p ∧ q.

The negation of p and q is

~ p : Delhi is not in India.

~ q : London is not in England.

And negation of p ∧ q is

~ (p ∧ q): Delhi is not in India or London is not in England.

(b) Let p : 3 + 4 = 7 and q : 8 < 11. The conjunction of statement p and q is p ∧ q.

~ p : 3 + 4 ≠ 7, ~ q : 8 ≮ 11.

And negation of p ∧ q is

~ (p ∧ q): 3 + 4 ≠ 7 or 8 ≮ 11.

1.15.2 Negation of Disjunction:

The negation of a disjunction p ∨ q is the conjunction of negation of p and negation of q. Mathematically, we write

$$\sim (p \vee q) = \sim p \wedge \sim q$$

The truth table of the above negation is

p	q	(p ∨ q)	~(p ∧ q)	~ p	~ q	~ p ∧ ~ 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

Example 1.27 Write the negation of both of following disjunction:

(a) Anjay is in class XI or Veronica is in class XII.

(b) 9 > 4 or 6 < 8.

Solution. (a) Let p : Anjay is in class XI and q : Veronica is in class XII.

The disjunction of statement p and q is p ∨ q.

The negation of p and q is

~ p : Anjay is not in class XI.

~ q : Veronica is not in class XII.

And negation of p ∨ q is

~ (p ∨ q): Anjay is not in class XI and Veronica is not in class XII.

(b) Let p : 9 > 4 and q : 6 < 8.

The disjunction of statement p and q is p ∨ q.

$\sim p : 9 \not> 4, \quad \sim q : 6 \not< 8.$

And negation of $p \vee q$ is

$\sim (p \vee q) : \ 9 \not> 4$ and $6 \not< 8$.

1.15.3 Negation of Conditional statement:

If p and q are two statements such that $p \rightarrow q$, then its negation is

$\sim (p \rightarrow q) = p \wedge \sim q$ as is verified by the following truth table:

p	q	$p \rightarrow q$	$\sim(p \rightarrow q)$	$\sim q$	$p \wedge \sim q$
T	T	T	F	F	F
T	F	F	T	T	T
F	T	T	F	F	F
F	F	T	F	T	F

Example 1.28 Write the negation of both of following conditional statements:

(a) If it is raining, then the game is cancelled.

(b) If he studies, he will pass the examination.

Solution. (a) Let p: It is raining and q: The game is cancelled.

Symbolically the statement (a) can be written as $p \rightarrow q$ and its negation is given by

$\sim(p \rightarrow q) = p \wedge \sim q$: It is raining and the game is not cancelled.

(b) The solution can be written on similar lines.

1.15.4 Negation of Biconditional statement:

If p and q are two statements such that $p \leftrightarrow q$, then its negation is

$\sim (p \leftrightarrow q) = p \leftrightarrow \sim q = \sim p \leftrightarrow q$ as is verified by the following truth table:

p	q	$p \leftrightarrow q$	$\sim(p \leftrightarrow q)$	$\sim p$	$\sim q$	$p \leftrightarrow \sim q$	$\sim p \leftrightarrow q$
T	T	T	F	F	F	F	F
T	F	F	T	F	T	T	T
F	T	F	T	T	F	T	T
F	F	T	F	T	T	F	F

Example 1.29 Write the negation the following Biconditional statement:

(a) He swims if and only if water is warm.

Solution. (a) Let p: He swims and q: water is warm.

Symbolically the statement (a) can be written as $p \leftrightarrow q$ and its negation is given by

$\sim (p \leftrightarrow q) = p \leftrightarrow \sim q = \sim p \leftrightarrow q$ = He swims if and only if water is not warm

= He does not swim if and only if water is warm.

(**Note: Negation of Negation**

The negation of negation of a statement is statement itself. If p is any statement, then we write $\sim (\sim p) = p$.

Example 1.30 Verify the statement

p : Roses are red.

Solution. The negation of statement is given by

~ p: Roses are not red. Further the negation of negation of statement p is

~ (~ p) : It is not the case that Roses are red.

Or It is false that roses are not red.

Roses are red.

1.16 Rule of Inference

1.17 Quantifiers

In the present chapter up to now we have studied compound statements joined by connectives $\sim, \rightarrow, \wedge, \vee$ and $\leftrightarrow$. It is not possible to use this study to determine validity in the majority of everyday and mathematical situations. For instance, the argument

All men are mortal,

All teachers are men

$\therefore$ All teachers are mortal

is intuitively correct. The methods studied so for are not able to derive the validity of this argument. To solve this purpose one has to separate the statement into parts, subjects and predicates.

1.17.1 Definition: Predicate Calculus and Statements Calculus

The symbolic analysis of predicates and quantified statements is called **predicate calculus** whereas the symbolic analysis of ordinary compound statements is called **statements calculus.**

In English grammar, predicate is the part of sentence that gives information about the subject. For example, in the sentence "Vizender is resident of Hisar", the word Vizender is subject and the phrase "is a resident of Hisar" is predicate. Thus, predicate is the part of sentence from which the subject has been removed.

In the logic, predicate can be obtained by removing any noun from statement. For example, if P stands for "is a resident of Hisar" and Q stands for " is a resident", then both P and Q are predicate symbols. The sentence "x is resident of Hisar" and " x is a resident of y" are denoted as P(x) and Q(x, y), respectively, where x and y are predicate variable that takes values in appropriate sets.

1.17.2 Definition: Predicate or propositional functions

A predicate is a sentence that contains a finite numbers of variables and becomes a statement when specific values of variables are substituted.

The domain of predicate variable is the set of all values that may be substituted in place of the variables. The predicates are also known as "propositional functions or open sentences".

1.17.3 Definition: Truth set of a predicate statement

Let P(x) be a predicate and x has domain D, then the set

$$\{ x \in D : P(x) \text{ is true} \}$$

is called **truth set** of P(x).

For example, let P(x) be " x is an integer less than 8" and suppose domain of x is set of all positive integers. Then the **truth set** of P(x) is {1, 2, 3, 4, 5, 6, 7}.

Let P(x) and Q(x) be predicates with common domain D of x. The notation $P(x) \Rightarrow Q(x)$ means every element in the truth set of P(x) is in the truth set of Q(x).

Similarly, $P(x) \Leftrightarrow Q(x)$ means P(x) and Q(x) has identical truth sets.

For instance, let domain of x be the set of positive integers. Let

P(x) : " x is integer less than 8"

Q(x) : "x is factor of 4"

Then,

Truth set of P(x) is {1, 2, 3, 4, 5, 6, 7} and the truth set of Q(x) is {1, 2, 4}.

Since every element in truth set of Q(x) is in truth set of P(x), so $Q(x) \Rightarrow P(x)$.

1.17.4 Definition: Quantifiers

The word that refer to quantities such as "All", or " Some" and tell for how many elements a given predicate is true are called quantifiers. By adding quantifier we can obtain statement from a predicate.

1.18 Types of Quantifiers

There are two types of quantifiers: (a) Universal Quantifier (b) Existential Quantifier

1.18.1 Definition: Universal Quantifier

The symbol ∀ denotes "for all" is Universal quantifier.

Thus the sentence, all men are strong, can be written as, ∀ x ∈ S, x is strong, where S denotes the set of all men.

1.18.2 Definition: Existential Quantifier

The symbol ∃ denotes "there exist" and is called **existential quantifier.** For example, the sentence " There is a University in Hisar" can be written as ∃ a University u such that u is in Hisar.

Or we can express

∃ u in U such that u is in Hisar, where U is the set of University.

The word "such that" is inserted just before the predicate.

1.19 Universal Statement and Existential Statement

1.19.1 Definition: Universal Statement

Let P(x) be predicate and D be domain of x. A statement of the form " ∀ x ∈ D" is called **universal statement**, which is true if and only if P(x) is true for every x in D and is false if and only if P(x) is false for at least one x in D. The value for x for which P(x) is false is called **Counterexample** to universal statement.

Example 1.31 Let D = {1, 2, 3, 4} and consider the universal statement, P(x): ∀ x ∈ D, $x^3 \geq x$. This is true for all values of x ∈ D since $1^3 \geq 1$, $2^3 \geq 2$ and so on. But the universal statement, Q(x): ∀ n ∈ N, n + 2 > 8 is not true because if we take n = 6, then 8 > 8 which is absurd.

1.19.2 Definition: Existential Statement

Let P(x) be predicate and D the domain of x. A statement of form " ∃ x in D such that P(x)" is called **existential statement**. It is defined to be true if and only if P(x) is true for at least one x in D and false if and only if P(x) is false for all x in D.

Example 1.32 (i) Consider the existential statement

∃ n ∈ N : n + 3 < 9

This is true since the set

$\{n : n + 3 < 9\} = \{1, 2, 3, 4, 5\}$ is not empty.

(ii) Let $A = \{2, 3, 4, 5\}$, then the existential statement

$\exists\ n \in A : n^2 = n$

is false because there is no element in A whose square is equal to itself.

1.20 Negation of Universal Statement

Definition: The negation of universal statement "$\forall\ x \in D, P(x)$" is logically equivalent to statement of form

$\exists$ x in D such that $\sim P(x)$.

Thus

$\sim (\forall\ x \in D, P(x)) = \exists$ x in D , $\sim P(x)$.

Hence

The negation of universal statement "all are" is logically equivalent to existential statement "some are not".

Example 1.33 Write the negation of universal statements

(a) $\forall\ n \in N, n + 2 > 9$ **Ans.** $\exists\ n \in N, n + 2 \leq 9$

(b) All students are intelligent. **Ans.** Some students are not intelligent.

(c) No politicians are honest. **Ans.** Some politicians are honest.

1.21 Universal Conditional Statement and its negation

1.21.1 Definition: Universal Conditional Statement

Statement of the form

$$\forall\ x, \text{ if } P(x) \text{ then } Q(x)$$

is called **universal conditional statement**.

For example, consider the statement

$$\forall\ x \in R, \text{ if } x > 2, \text{ then } x^3 > 8$$

is universal conditional statement.

1.21.2 Negation of Universal Conditional Statement

Definition: The negation of a **universal conditional statement** is defined by

$$\sim (\forall\ x, \text{ if } P(x) \rightarrow Q(x)) = \exists\ x \text{ such that } \sim (P(x) \rightarrow Q(x)).$$

As we know that the negation of if- then statement is

$\sim (P(x) \rightarrow Q(x)) = \exists$ x such that $P(x) \wedge \sim Q(x)$.

Hence

$\sim (\forall$ x, if $P(x) \rightarrow Q(x)) = \exists$ x such that $P(x) \wedge \sim Q(x)$,

That is

$\sim (\forall$ x, if $P(x) \rightarrow Q(x)) = \exists$ x such that P(x) and $\sim Q(x)$.

Example 1.34 Write the negation of universal conditional statements

"For all human being x, if x is man, then x is strong".

Solution. The negation of given universal conditional statement is

"$\exists$ a human being x such that x is man and x is not strong".

1.22 Contrapositive, Converse and Inverse of Universal Conditional Statement

Definition: Let

$\forall$ x $\in$ D, if P(x) then Q(x) be a statement. Then

(i) **Contrapositive** of the statement is

$\forall$ x $\in$ D, if $\sim Q(x)$ then $\sim P(x)$.

(ii) **Converse** of statement is

$\forall$ x $\in$ D, if Q(x) then P(x).

(iii) **Inverse** of statement is

$\forall$ x $\in$ D, if $\sim P(x)$ then $\sim Q(x)$.

1.23 Check Your Progress

1. Consider the quantified statement: "Every student has at least one course where the lecturer is a teaching assistant." Its negation is the statement:---------

2. Find the truth table of $\neg p \wedge q$.

3. Rewrite the following statements without using the conditional:

(a) If it is cold, he wears a hat.

(b) If productivity increases, then wages rise.

4. Determine the contrapositive of each statement:

(a) If Erik is a poet, then he is poor.

(b) Only if Marc studies will he pass the test.

5. Let A = {1, 2, 3, 4, 5}. Determine the truth value of each of the following statements:

(a) $(\exists x \in A)(x + 3 = 10)$ (c) $(\exists x \in A)(x + 3 < 5)$

(b) $(\forall x \in A)(x + 3 < 10)$ (d) $(\forall x \in A)(x + 3 \leq 7)$

Summary

In this chapter, we have discussed about the Logical statements, their symbolic representations, construction of truth tables, laws of algebra of Propositions and Quantifiers. Examples are also presented related to these topics.

1.24 Keywords

Propositions, logical operations, logical equivalence, Quantifiers

1.25 Self-Assessment Test

1. What is the negation of each of the following proposition ?

(i) Today is Tuesday

(ii) A cow is an animal

(iii) No one wants to buy my house

(iv) Some people have no scooter

2. Determine the truth value of each of the following statements

(i) 6+2=7 and 4+4=8

(ii) 3+1=4 and 4+5=7

(iii) 4+3=7 and 6+2=8

(iv) 2+3=5 and 3+1=2

3. Find the principal conjunctive normal form without using truth table.

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

4. The principle disjunctive normal form of A is $(p \wedge q) \vee (\sim p \wedge r) \vee (q \wedge r)$, find principal conjunctive.

5. Show that the proposition $\wedge \sim q$ and $(p \vee q) \wedge (\sim p) \wedge (\sim q)$ are contradiction.

1.26 Answers to check your progress

1. There is a student such that in every course the lecturer is not a teaching assistant.

2.

P	q	¬p	¬p ∧ q
T	T	F	F
T	F	F	F
T	T	T	T
T	F	T	F

3. (a) It is not cold or he wears a hat.

(b) Productivity does not increase or wages rise.

4. (a) The contrapositive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$. Hence the contrapositive follows: If Erik is not poor, then he is not a poet.

(b) The statement is equivalent to: "If Marc passes the test, then he studied." Thus its contrapositive is: If Marc does not study, then he will not pass the test.

5. (a) False. For no number in A is a solution to $x + 3 = 10$.

(b) True. For every number in A satisfies $x + 3 < 10$.

(c) True. For if $x0 = 1$, then $x0 + 3 < 5$, i.e., 1 is a solution.

(d) False. For if $x0 = 5$, then $x0 + 3$ is not less than or equal 7. In other words, 5 is not a solution to the given condition.

1.27 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.

2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.

3 C.L. Liu, Elements of Discrete Mathematics, McGraw-Hili Book Co.

4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 2 **Written by Dr. Vizender Singh**

LATTICES

Structure:

2.0 Learning Objectives

In this chapter, the reader will learn about basic concepts of lattice, lattices as partially ordered sets, their various properties, lattices as algebraic systems and sub lattices. Examples are also given related to these topics.

2.1 Introduction

In this course we will consider mathematical objects known as lattices. What is a lattice? It is a set of points in n-dimensional space with a periodic structure, such as the one illustrated in Figure 1. Three dimensional lattices occur naturally in crystals, as well as in stacks of oranges. Historically, lattices were investigated since the late 18th century by mathematicians such as Lagrange, Gauss, and later Minkowski. More recently, lattices have become a topic of active research in computer science. They are used as an algorithmic tool to solve a wide variety of problems; they have many applications in cryptography and cryptanalysis; and they have some unique properties from a computational complexity point of view. These are the topics that we will see in this course.

2.2 Definition of Lattice and Related Examples

Definition: A **lattice** is a partially ordered set (L, ≤) in which every subset {a, b} consisting of two elements has a **least upper bound** and a **greatest lower bound.**

We denote LUB ({a, b}) by a ∨ b and call it **join or sum of a and b**. Similarly, we denote GLB ({a, b}) by a ∧ b and call it **meet or product of a and b.**

Other symbols used are:

LUB : $\oplus$, + , $\cup$

GLB : *, · , $\cap$

Thus **Lattice** is a mathematical structure with **two binary operations, join and meet**. Lattice structures often appear in computing and mathematical applications.

A totally ordered set is obviously a lattice but not all partially ordered sets are lattices.

Example 1: Let A be any set and P(A) be its power set. The partially ordered set (P(A), $\subseteq$) is a lattice in which the meet and join are the same as the operations $\cap$ and $\cup$ respectively. If A has single element, say a, then P(A) = {φ, {a}} and

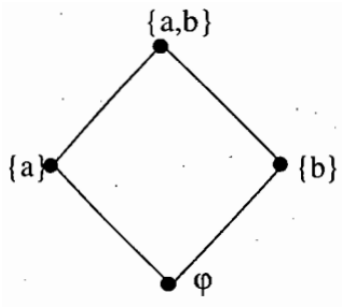
LUB ({φ, {a}) = {a}

GLB ({φ, {a}) = φ

The Hasse diagram of (P(A), $\subseteq$) is a chain containing two elements φ and {a} as shown below:

If A has two elements, say a and b.

Then P(A) = {φ, {a}, {b}, {a, b}}.The Hasse diagram of (P(A), $\subseteq$) is then as shown below :



We note that

1. LUB exists for every two subsets and is L $\cup$ M for L, M $\in$ P(A).

2. GLB exists for every two subsets and is L $\cap$ M for L, M $\in$ P(A).

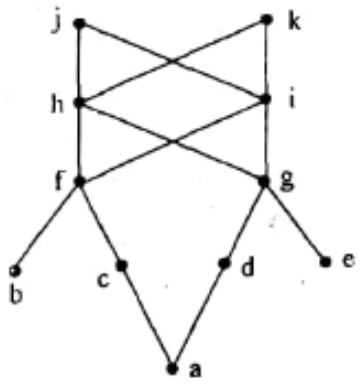
Hence (P(A), $\subseteq$) is a lattice.

Example 2: Consider the poset (**N**, $\leq$), where $\leq$ is relation of divisibility. Then **N** is a lattice in which

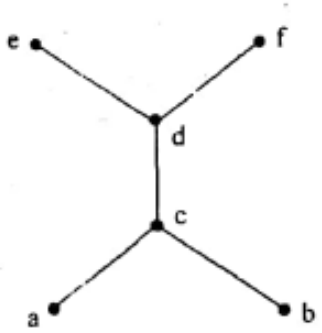
join of a and b = a $\vee$ b = LCM (a, b)

meet of a and b = a $\wedge$ b = GCD (a, b) for a, b $\in$ N.

Example 3: Which of the following Hasse diagram represent lattices?



(i)



(ii)

Solution: The Hasse diagram (i) is not a lattice because $c \vee d$ and $b \wedge c$ do not exist. The Hasse diagram (ii) is not a lattice because $a \wedge b$ does not exist, $e \vee f$ does not exist.

2.3 Dual Lattice

Definition: Let $(L, \leq)$ be a poset and let $(L, \geq)$ be the dual poset. If $(L, \leq)$ is a lattice, then $(L, \geq)$ is also a lattice. In fact, for any a and b in L, the LUB of a and b in $(L, \leq)$ is equal to the GLB of a and b in $(L, \geq)$. Similarly, the GLB of a and b in $(L, \leq)$ is equal to LUB of a and b in $(L, \geq)$.
The operation $\vee$ and $\wedge$ are called dual of each other.

Example: Let S be a set and L = P(S), where P(S) is the power set of S. Then $(L, \subseteq)$ is a lattice and its dual lattice is $(L, \supseteq)$, where $\supseteq$ represents "contains". We note that in the poset $(L, \supseteq)$, the join $A \vee B$ is the set $A \cap B$ and the meet $A \wedge B$ is the set $A \cup B$.

2.4 Cartesian Product of Lattices

Theorem: If $(L_1, \leq_1)$ and $(L_2, \leq_2)$ are lattices, then $(L, \leq)$ is a lattice, where $L = L_1 \times L_2$ and the partial order $\leq$ of L is the product partial order.

Proof: We denote the join and meet in L_1 by $\vee_1$ and $\wedge_1$ and the join and meet in L_2 by $\vee_2$ and $\wedge_2$ respectively. We know that Cartesian product of two posets is a poset. Therefore $L = L_1 \times L_2$ is a poset.
Thus all we need to show is that if (a_1, b_1) and $(a_2, b_2) \in L$, then $(a_1, b_1) \vee (a_2, b_2)$ and $(a_1, b_1) \wedge (a_2, b_2)$ exist in L.

Further, we know that

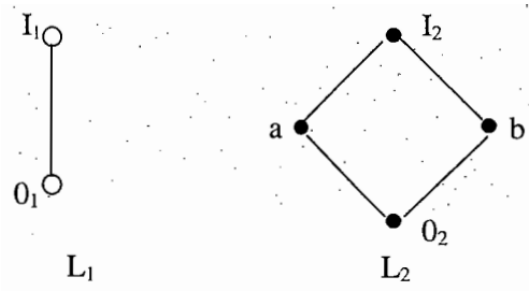
$$(a_1, b_1) \vee (a_2, b_2) = (a_1 \vee_1 a_2, b_1 \vee_2 b_2)$$

and

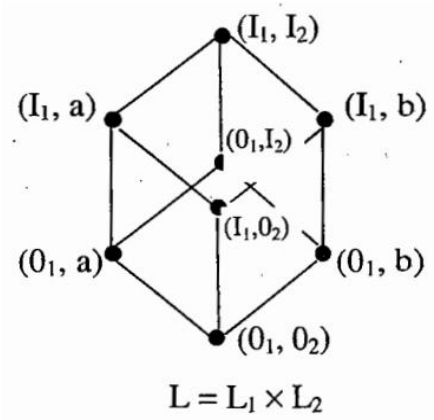
$$(a_1, b_1) \wedge (a_2, b_2) = (a_1 \wedge_1 a_2, b_1 \wedge_2 b_2)$$

Since L_1 is lattice, then $a_1 \vee_1 a_2$ and $a_1 \wedge_1 a_2$ exist in L_1. Similarly, since L_2 is a lattice, then $b_1 \vee_2 b_2$ and $b_1 \wedge_2 b_2$ exist in L_2. Hence $(a_1, b_1) \vee (a_2, b_2)$ and $(a_1, b_1) \wedge (a_2, b_2)$ both exist and therefore $(L, \leq)$ is a lattice, called **the direct product of $(L_1, \leq_1)$ and $(L_2, \leq_2)$.**

Example: Let L_1 and L_2 be the lattices whose Hasse diagrams are given below:



Then $L = L_1 \times L_2$ is the lattice shown in the diagram below:



2.5 Properties of Lattices

Let $(L, \leq)$ be a lattice and let $a, b, c \in L$. Then, from the definition of $\vee$ (join) and $\wedge$ (meet), we have

(i) $a \le a \vee b$ and $b \le a \vee b$; $a \vee b$ is an upper bound of a and b.

(ii) if $a \le c$ and $b \le c$, then $a \vee b \le c$; $a \vee b$ is the least upper bound of a and b.

(iii) $a \wedge b \le a$ and $a \wedge b \le b$; $a \wedge b$ is a lower bound of a and b.

(iv) if $c \le a$ and $c \le b$, then $c \le a \wedge b$; $a \wedge b$ is the greatest lower bound of a and b.

2.5.1 Theorem: Let L be a lattice. Then for every a and b in L,

(i) $a \vee b = b$ if and only if $a \le b$

(ii) $a \wedge b = a$ if and only if $a \le b$

(iii) $a \wedge b = a$ if and only if $a \vee b = b$

Proof: (i) Let $a \vee b = b$.

Since $a \le a \vee b$, we have $a \le b$.

Conversely, if $a \le b$, then since $b \le b$, it follows that b is an upper bound of a and b. Therefore, by the definition of least upper bound, we have

$$a \vee b \le b \qquad (1)$$

Also $a \vee b$ being an upper bound, we have

$$b \le a \vee b \qquad (2)$$

Hence from (1) and (2), we obtain $a \vee b = b$.

(ii) Let $a \wedge b = a$.

Since $a \wedge b \le b$, we have $a \le b$.

Conversely, if $a \le b$ and since $a \le a$, it follows that a is a lower bound of a and b. So, by the definition of greatest lower bound, we have

$$a \le a \wedge b \qquad (3)$$

Since $a \wedge b$ is lower bound, we have

$$a \wedge b \leq a \qquad (4)$$

Hence from (3) and (4), we obtain

$$a \wedge b = a.$$

(iii) From (ii), we have

$$a \wedge b = a \Leftrightarrow a \leq b \qquad (5)$$

From (i), we have

$$a \leq b \Leftrightarrow a \vee b = b \qquad . \qquad (6)$$

Hence, combining (5) and (6), we have

$$a \wedge b = a \Leftrightarrow a \vee b = b.$$

Example: Let L be a linearly (total) ordered set. Therefore $a, b \in L$ imply either $a \leq b$ or $b \leq a$. Therefore, the above theorem implies that

$$a \vee b = a$$

and $a \wedge b = a$

Thus for every pair of elements a, b in L, $a \vee b$ and $a \wedge b$ exist. Hence **a linearly ordered set is a lattice.**

2.5.2 Theorem: Let $(L, \leq)$ be a lattice and let $a, b, c \in L$. Then we have

L_1 : Idempotent property

(i) $a \vee b = a$

(ii) $a \wedge b = a$

L_2 : Commutative property

(i) $a \vee b = b \vee a$

(ii) $a \wedge b = b \wedge a$

L_3 : Associative property

(i) $a \vee (b \vee c) = (a \vee b) \vee c$

(ii) $a \wedge (b \wedge c) = (a \wedge b) \wedge c$

L_4 : Absorption property

(i) $a \vee (a \wedge b) = a$

(ii) $a \wedge (a \vee b) = a$

Proof: L_1 : The idempotent property follows from the definition of LUB and GLB.

L_2: Commutativity follows from the symmetry of a and b in the definition of LUB and GLB.

L_3: (i) From the definition of LUB, we have

$$a \leq a \vee (b \vee c) \qquad (1)$$

and $$b \vee c \leq a \vee (b \vee c) \qquad (2)$$

Also

$$b \leq b \vee c \qquad (3)$$

and

$$c \leq b \vee c \qquad (4)$$

Then from (2) and (3) and using transitivity property, we have

$$b \leq a \vee (b \vee c) \qquad (5)$$

Also from (2) and (4) and using transitivity property, we have

$c \leq a \vee (b \vee c)$ (6)

Now, (1) and (5) imply that $a \vee (b \vee c)$ is an upper bound of a and b and hence by the definition of least upper bound, we have

$a \vee b \leq a \vee (b \vee c)$ (7)

Also by (6) and (7), we have $a \vee (b \vee c)$ is an upper bound of c and $a \vee b$. Therefore

$(a \vee b) \vee c \leq a \vee (b \vee c)$ (8)

Similarly, we have

$a \vee (b \vee c) \leq (a \vee b) \vee c$ (9)

Hence, by antisymmetry of the relation $\leq$, (8) and (9) yield

$a \vee (b \vee c) = (a \vee b) \vee c$

The proof of (ii) is analogous to the proof of part (i).

L_4: (i) Since $a \wedge b \leq a$ and $a \leq a$, it follows that a is an upper bound of $a \wedge b$ and a. Therefore, by the definition of least upper bound, we have

$a \vee (a \vee b) \leq a$ (10)

On the other hand, by the definition of LUB, we have

$a \leq a \vee (a \wedge b)$ (11)

The expression (10) and (11) yields

$a \vee (a \wedge b) = a.$

(ii) Since $a \leq a \vee b$ and $a \leq a$, it follows that a is a lower bound of $a \vee b$ and a. Therefore, by the definition of GLB, we have

$a \leq a \wedge (a \vee b)$ (12)

Also, by the definition of GLB, we have

$$a \wedge (a \vee b) \leq a \qquad (13)$$

Then (12) and (13) imply

$$a \wedge (a \vee b) = a$$

Hence the result.

Note: In view of L_3, we can write $a \vee (b \vee c)$ and $(a \vee b) \vee c$ as $a \vee b \vee c$.

Thus, we can express

LUB $(\{a_1, a_2, \ldots, a_n\})$ as $a_1 \vee a_2 \vee \ldots\ldots \vee a_n$

GLB $(\{a_1, a_2, \ldots, a_n\})$ as $a_1 \wedge a_2 \wedge \ldots\ldots \wedge a_n$

Remark: Using commutative and absorption properties, part (ii) of Theorem 2.5.1 can be proved as follows:

Let $a \wedge b = a$.

Then we have

$$b \vee (a \wedge b) = b \vee a$$

$$= a \vee b \qquad \text{(Commutativity)}$$

$$\therefore \quad b \vee (a \wedge b) = a \vee b \qquad (1)$$

But

$$b \vee (a \wedge b) = b \qquad \text{(Absorption property)} \qquad (2)$$

Hence from (1) and (2), we have

$$a \vee b = b$$

So by part (i) of Theorem 2.5.1, we have $a \leq b$.

Hence $a \wedge b = a$ if and only if $a \leq b$.

2.5.3 Theorem: Let $(L, \leq)$ be a lattice. Then for any $a, b, c \in L$, the following properties hold:

1. **(Isotonicity):** If $a \leq b$, then

(i) $a \vee c \leq b \vee c$

(ii) $a \wedge c \leq b \wedge c$

This property is called "Isotonicity".

2. $a \leq c$ and $b \leq c$ if and only if $a \vee b \leq c$

3. $c \leq a$ and $c \leq b$ if and only if $c \leq a \wedge b$

4. If $a \leq b$ and $c \leq d$, then

(i) $a \vee c \leq b \vee d$

(ii) $a \wedge c \leq b \wedge d$

Proof: 1 (i).We know that

$a \vee b = b$ if and only if $a \leq b$.

Therefore, to show that $a \vee c \leq b \vee c$, we shall show that

$(a \vee c) \vee (b \vee c) = b \vee c$

Now we have

$$(a \vee c) \vee (b \vee c) = [(a \vee c) \vee b] \vee c$$

$$= a \vee (c \vee b) \vee c$$

$$= a \vee (b \vee c) \vee c$$

$$= (a \vee b) \vee (c \vee c)$$

$= b \vee c$ ($\because a \vee b = b$ and $c \vee c = c$)

The part 1 (ii) can be proved similarly.

2. If $a \leq c$, then 1(i) implies

$$a \vee b \leq c \vee b$$

But

$$b \leq c \Leftrightarrow b \vee c = c$$

$\Leftrightarrow c \vee b = c$ (using commutativity)

Hence $a \leq c$ and $b \leq c$ if and only if $a \vee b \leq c$

3. If $c \leq a$, then 1(ii) implies

$$c \wedge b \leq a \wedge b$$

But

$$c \leq b \Leftrightarrow c \wedge b = c$$

Hence $c \leq a$ and $c \leq b$ if and only if $c \leq a \wedge b$.

4 (i) We note that 1(i) implies that

if $a \leq b$, then $a \vee c \leq b \vee c = c \vee b$

if $c \leq d$, then $c \vee b \leq d \vee b = b \vee d$

Hence, by transitivity, we have

$$a \vee c \leq b \vee d$$

(ii) We note that l (ii) implies that

if $a \leq b$, then $a \wedge c \leq b \wedge c = c \wedge b$

if $c \leq d$, then $c \wedge b \leq d \wedge b = b \wedge d$

Therefore, transitivity implies

$a \wedge c \leq b \wedge d$

2.5.4 Distributive Inequalities

Theorem: Let $(L, \leq)$ be a lattice. If $a, b, c \in L$, then

(1) $a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$

(2) $a \wedge (b \vee c) \geq (a \wedge b) \vee (a \wedge c)$

Proof: (1) We have

$a \leq a \vee b$ and $a \leq a \vee c$ (i)

We know the result " if $x \leq y$ and $x \leq z$ in a lattice, then $x \leq y \wedge z$ ". (*)

Using the above result, then (i) yields

$a \leq (a \vee b) \wedge (a \vee c)$ (ii)

Also we have

$b \wedge c \leq b \leq a \vee b$ (iii)

and

$b \wedge c \leq c \leq a \vee c$ (iv)

Then from (iii) and (iv), we have

$b \wedge c \leq a \vee b$ and $b \wedge c \leq a \vee c$ (v)

Using the result (*), then (v) yields

$b \wedge c \leq (a \vee b) \wedge (a \vee c)$ (vi)

Also, again by the above theorem if $x \leq z$ and $y \leq z$ in a lattice, then

$$x \vee y \leq z \qquad (**)$$

Using the result (**), then (ii) and (vi) yield

$$a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$$

This proves (1).

The second distributive inequality follows by using the principle of duality.

2.5. Modular Inequality

Theorem: Let $(L, \leq)$ be a lattice. If $a, b, c \in L$, then

$$a \leq c \text{ if and only if } a \vee (b \wedge c) \leq (a \vee b) \wedge c$$

Proof: We know that

$$a \leq c \Leftrightarrow a \vee c = c \qquad (1)$$

Also, by distributive inequality,

$$a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c) \qquad (2)$$

Therefore using (1) in (2), we have

$$a \leq c \text{ if and only if } a \vee (b \wedge c) \leq (a \vee b) \wedge c$$

This proves the result.

The modular inequalities can be expressed in the following way also:

$$(a \wedge b) \vee (a \wedge c) \leq a \wedge [b \vee (a \wedge c)]$$

and $$(a \vee b) \wedge (a \vee c) \geq a \vee [b \wedge (a \vee c)]$$

Example: Let $(L, \leq)$ be a lattice and $a, b, c \in L$. If $a \leq b \leq c$, then $(a \wedge b) \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

Solution: Since $a \leq b$ and $b \leq c$, we have

$$a \wedge b = a \quad \text{and} \quad b \wedge c = b \qquad (1)$$

Thus

$$(a \wedge b) \vee (b \wedge c) = a \vee b \qquad \text{[using (1)]}$$

$$= b \qquad \text{[since } a \leq b \Leftrightarrow a \vee b = b\text{]} \qquad (2)$$

Also by transitivity, we have

$$a \leq b \leq c \Rightarrow a \leq c$$

Then

$$a \leq b \text{ and } a \leq c \text{ implies } a \vee b = b \text{ and } a \vee c = c \qquad (3)$$

Now we have

$$(a \vee b) \wedge (a \vee c) = b \wedge c \qquad \text{[using (3)]}$$

$$= b \quad \text{[since } b \leq c \Leftrightarrow b \wedge c = b\text{]} \qquad (4)$$

Hence from (2) and (4), we have

$$(a \wedge b) \vee (b \wedge c) = b = (a \vee b) \wedge (a \vee c)$$

Thus $(a \wedge b) \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

This proves (ii).

2.6 Lattices as Algebraic System

Definition: A **Lattice** is an algebraic system $(L, \vee, \wedge)$ with two binary operations $\vee$ and $\wedge$, called **join** and **meet** respectively, on a non-empty set L which satisfy the following axioms for $a, b, c \in L$:

1. Commutative Law:

$a \vee b = b \vee a$ and $a \wedge b = b \wedge a$

2. Associative Law:

$$(a \vee b) \vee c = a \vee (b \vee c)$$

and

$$(a \wedge b) \wedge c = a \wedge (b \wedge c)$$

3. Absorption Law:

(i) $a \vee (a \wedge b) = a$

(ii) $a \wedge (a \vee b) = a$

We observe that Idempotent Law follows from axiom 3 above.

In fact, we have

$a \vee a = a \vee [a \wedge (a \vee b)]$ [using 3(ii)]

$= a$ [using 3(i)]

The proof of $a \wedge a = a$ follows by principle of duality.

2.7 Partial Order Relation on a Lattice

A partial order relation on a lattice (L) follows as a consequence of the axioms for the binary operations $\vee$ and $\wedge$.

We define a relation $\leq$ on L such that for $a, b \in L$,

$$a \leq b \Leftrightarrow a \vee b = b$$

or analogously,

$$a \leq b \Leftrightarrow a \wedge b = a$$

Now we have

(i) For any $a \in L$,

$a \vee a = a$ (using idempotent law)

Therefore, we have $a \leq a$

This shows that the relation $\leq$ is reflexive.

(ii) Let $a \leq b$ and $b \leq a$. Therefore

$a \vee b = b$

and $b \vee a = a$

But we know that

$a \vee b = b \vee a$ (using Commutative Law in lattice)

Hence

$a = b$

This shows that the relation $\leq$ is antisymmetric.

(iii) Suppose that $a \leq b$ and $b \leq c$.

Therefore

$a \vee b = b$ and $b \vee c = c$ (1)

Now we have

$a \vee c = a \vee (b \vee c)$ [using (1)]

$= (a \vee b) \vee c$ (using Associativity in lattice)

$= b \vee c$ [using (1)]

$= c$ [using (1)]

$\Rightarrow$ $a \vee c = c$

So we have, $a \leq c$

Hence the relation $\leq$ is transitive.

Hence the relation $\leq$ is a partial order relation on a lattice L.

This shows that a **lattice is a partially ordered set.**

2.8 Least Upper Bounds and Greatest Lower Bounds in a Lattice

Let $(L, \vee, \wedge)$ be a lattice and let $a, b \in L$. We now show that LUB of $\{a, b\} \subseteq L$ with respect to the partial order relation introduced in section 2.7 is $a \vee b$ and GLB of $\{a, b\}$ is $a \wedge b$.

From absorption law,

$$a \wedge (a \vee b) = a$$

and $$b \wedge (a \vee b) = b$$

Therefore $a \leq a \vee b$ and $b \leq a \vee b$, showing that $a \vee b$ is an upper bound for $\{a, b\}$. Suppose that there exists $c \in L$ such that $a \leq c$, $b \leq c$. Thus we have

$$a \vee c = c \quad \text{and} \quad b \vee c = c \qquad (1)$$

Then we have

$$(a \vee b) \vee c = a \vee (b \vee c) \quad \text{(using Associativity in lattice)}$$

$$= a \vee c \quad [\text{using (1)}]$$

$$= c \quad [\text{using (1)}]$$

This implies that $a \vee b \leq c$

Hence $a \vee b$ is the least upper bound of a and b.

Similarly, we can show that $a \wedge b$ is GLB of a and b.

The above discussion shows that the two definitions of lattice given so far are equivalent.

2.9 Sublattices

Definition: Let $(L, \leq)$ be a lattice. A non-empty subset S of L is called a **sublattice** of L if $a \vee b \in S$ and $a \wedge b \in S$ whenever $a \in S, b \in S$.

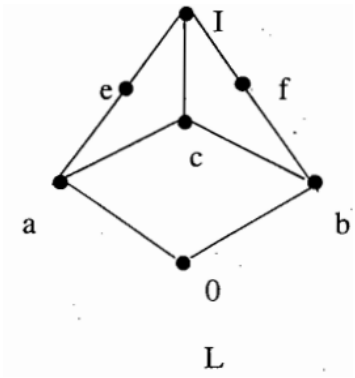
Or

Let $(L, \vee, \wedge)$ be a lattice and let $S \subseteq L$ be a subset of L. Then $(S, \vee, \wedge)$ is called a **sublattice** of $(L, \vee, \wedge)$ if and only if S is closed under both operations of join $(\vee)$ and meet $(\wedge)$.

From the definition it is clear that **sublattice itself is a lattice.**

However, **any subset of L which is a lattice need not be a sublattice.**

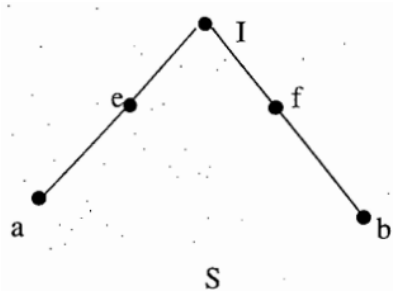
For example, consider the lattice shown in the diagram:



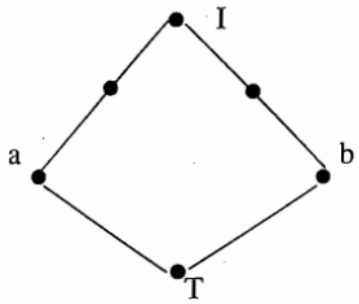
We see that

(i) The subset S shown by the diagram below is not a sublattice of L, since

$a \wedge b \notin S$ and $a \vee b \notin S$

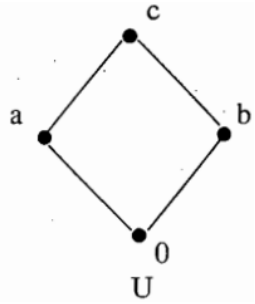


(ii) The set T shown below is not a sublattice of L since $a \vee b \notin T$.



However, T is a lattice when considered as a poset by itself.

(iii) The subset U of L shown below is a sublattice of L:



Example: Let A be any set and P(A) its power set. Then (P(A), $\vee$, $\wedge$) is a lattice in which join and meet are union of sets and intersection of sets respectively.

A family $\hat{C}$ of subsets of A such that S $\cup$ T and S $\cap$ T are in $\hat{C}$ for S, T $\in \hat{C}$ is a sublattice of (P(A), $\vee$, $\wedge$). Such a family $\hat{C}$ is called a ring of subsets of A and is denoted by (R(A), $\vee$, $\wedge$) (This is not a ring in the sense of algebra). Some author can it lattice of subsets.

2.10 Check Your Progress

1. Write the dual of each statement:

(a) $(a \wedge b) \vee c = (b \vee c) \wedge (c \vee a)$;

(b) $(a \wedge b) \vee a = a \wedge (b \vee a)$.

2. Fill in the Blanks

Let (L, $\leq$) be a lattice and a, b, c $\in$ L. If ----------then $a \vee b = b \wedge c$.

Sol: We know that

$$a \leq b \Leftrightarrow a \vee b = b$$

and

Hence $a \leq b \leq c$ implies

$a \vee b = b \wedge c$.

3. Let n be a positive integer and let D_n be the set of all positive divisors of n. Then D_n is a lattice under the relation of divisibility. The Hasse diagram of the lattices D_8, D_{20} and D_{30} are respectively.

4. The lattice $(D_n, \leq)$ is a ---------- of $(\mathbf{N}, \leq)$, where $\leq$ is the relation of divisibility.

5. Let $\hat{C}$ be collection of sets with binary operations Union and Intersection of sets. Then $(\hat{C}, \cup, \cap)$ is a lattice. In this lattice, the partial order relation is set inclusion. In fact, for A, B $\in \hat{C}$,

Or

$A \subseteq B$ iff $A \cap B = A$

For example, the diagram of lattice of subsets of {a, b} is

2.11 Summary

In this chapter, we have studied about lattices, its various properties and sublattices. Examples are also given related to these topics.

2.12 Keywords

Lattice, joint and meet operations, Hasse diagram, sublattices

2.13 Self-Assessment Test

1. Show that the following "weak" distributive laws hold for any lattice L:

(a) $a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$;

(b) $a \wedge (b \vee c) \geq (a \wedge b) \vee (a \wedge c)$

2. Consider the lattice N of positive integers ordered by divisibility.

(a) Which elements are join-irreducible ?

(b) Which elements are atoms?

3. Prove that every distributive lattice is modular.

4. Let R be a ring. Let L be the collection of all ideals of R. Prove that L is a bounded lattice where, for any ideals J and K of R, we define: $J \vee K = J + K$ and $J \wedge K = J \cap K$.

5. Let $S = \{1, 2, 3, 4\}$. We use the notation $[12, 3, 4] \equiv [\{1, 2\}, \{3\}, \{4\}]$.

Three partitions of S follow: P1 = [12, 3, 4], P2 = [12, 34], P3 = [13, 2, 4] then find the other twelve partitions of S

2.14 Answers to check your progress

1. Replace $\vee$ by $\wedge$ and replace $\wedge$ by $\vee$ in each statement to obtain the dual statement:

(a) $(a \vee b) \wedge c = (b \wedge c) \vee (c \wedge a)$;

(b) $(a \vee b) \wedge a = a \vee (b \wedge a)$

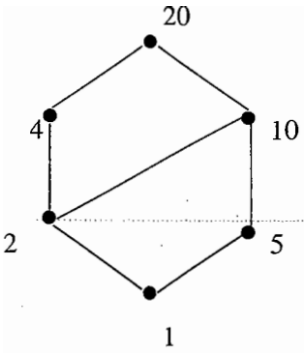
2. (i) $a \leq b \leq c$,

(ii) $b \leq c \Leftrightarrow b \wedge c = b$

3.(i)

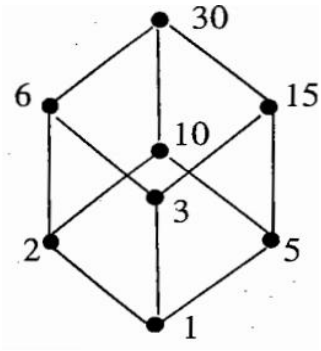
$D_8 = \{1, 2, 4, 8\}$

(ii)



$D_{20} = \{1, 2, 4, 5, 10, 20\}$

(iii)

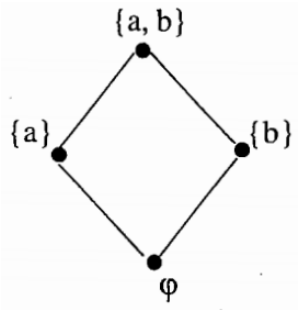


D_{30} = {1, 2, 3, 5, 6, 10, 15, 30}

4. Sublattice

5. (i) $A \subseteq B$ iff $A \cup B = B$

(ii)



2.15 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.
2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.
3 C.L. Liu, Elements of Discrete Mathematics, McGraw-Hili Book Co.
4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 3 **Written by Dr. Vizender Singh**

SOME SPECIAL LATTICES

Structure:

3.0 Learning Objectives

In the present chapter, the reader will learn about lattice isomorphism, some special lattices namely, complete lattice, bounded lattice, complemented lattice and distributive lattice. Further, the reader will learn about join- irreducible elements and atoms in a lattice.

3.1 Introduction

In the previous chapter, we have studied about lattices, their various properties, lattices as algebraic systems and sublattices. In the present chapter, we will discuss lattice isomorphism, some special lattices namely, complete lattice, bounded lattice, complemented lattice and distributive lattice. Examples are also given related to these special lattices. Further, we will discuss about join- irreducible elements and atoms in a lattice.

3.2 Lattice Isomorphism

3.2.1 Lattice Homomorphism

Definition: Let $(L_1, \vee_1, \wedge_1)$ and $(L_2, \vee_2, \wedge_2)$ be two lattices.

A mapping $f : L_1 \to L_2$ is called a **lattice homomorphism** from the lattice $(L_1, \vee_1, \wedge_1)$ to $(L_2, \vee_2, \wedge_2)$ if for any $a, b \in L_1$,

$$f(a \vee_1 b) = f(a) \vee_2 f(b)$$

and $$f(a \wedge_1 b) = f(a) \wedge_2 f(b)$$

Thus, here both the binary operations of join and meet are preserved. There may be mappings which preserve only one of the two operations. Such mappings are not lattice homomorphism.

Let $\leq_1$ and $\leq_2$ be partial order relations on $(L_1, \vee_1, \wedge_1)$ and $(L_2, \vee_2, \wedge_2)$ respectively. Let $f : L_1 \to L_2$ be lattice homomorphism. If $a, b \in L_1$, then

$$a \leq_1 b \Leftrightarrow a \vee_1 b = b$$

Then we have

$$f(b) = f(a \vee_1 b)$$

$$= f(a) \vee_2 f(b)$$

Thus $$f(b) = f(a) \vee_2 f(b) \Leftrightarrow f(a) \leq_2 f(b)$$

Thus

$a \leq_1 b \Leftrightarrow f(a) \leq_2 f(b)$

Thus **order relations are also preserved** under lattice homomorphism.

3.2.2 Lattice Isomorphism

Definition: If a lattice homomorphism $f : L_1 \to L_2$ is one-to-one and onto, then it is called **lattice isomorphism**.

If there exists an isomorphism between two lattices, then the lattices are called **isomorphic.**

Since lattice isomorphism preserves order relation, therefore **isomorphic lattices can be represented by the same diagram in which nodes are replaced by images.**

Theorem: Let $A = \{a_1, a_2,....,a_n\}$ and $B = \{b_1 , b_2,.....,b_n\}$ be any two finite sets with n elements. Then the lattices $(P(A), \subseteq)$ and $(P(B), \subseteq)$ are isomorphic and so have identical Hasse diagram.

Proof: Consider the mapping

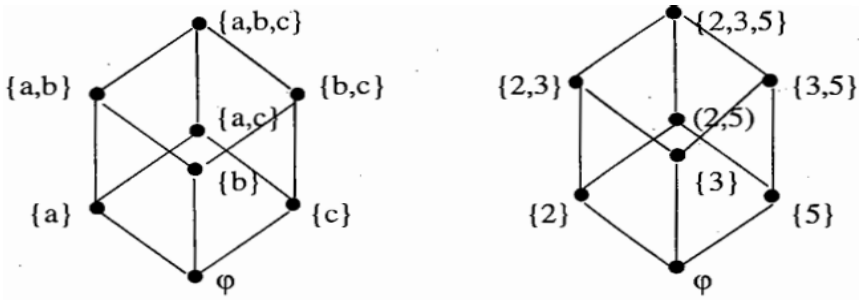
$$f : P(A) \to P(B)$$

defined by

$$f(\{a_n\}) = \{b_n\},\ f(\{a_1, a_2,....,a_m\}) = \{b_1, b_2,.....,b_n\} \text{ for } m \leq n$$

Then f is bijective mapping and $L \subseteq M \Leftrightarrow f(L) \subseteq f(M)$ for subsets L and M of P(A). Hence P(A) and P(B) are isomorphic.

For example, let A = {a, b, c}, B = {2, 3, 5}. The Hasse diagram of P(A) and P(B) are then given below:



Define a mapping f : P(A) $\rightarrow$ P(B) by

$f(\varphi) = \varphi$, $f(\{a\}) = \{2\}$, $f(\{b\}) = \{3\}$, $f(\{c\}) = \{5\}$,

$f(\{a, b\}) = \{2, 3\}$, $f(\{b, c\}) = \{3, 5\}$, $f(\{a, c\}) = \{2, 5\}$

and

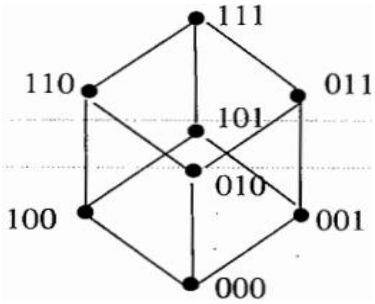
$f(\{a, b, c\}) = \{2, 3, 5\}$

This is a bijective mapping satisfying the condition that if S and T are subsets of A, then $S \subseteq T$ if and only if $f(S) \subseteq f(T)$.

Hence f is isomorphism and $(P(A), \subseteq)$ and $(P(B), \subseteq)$ are isomorphic.

Thus, for each n = 0, 1, 2, ..., there is only one type of lattice and this lattice depends only on n, the number of elements in the set A, and not on A. It has 2^n elements. Also, we know that if A has n elements, then all subsets of A can be represented by sequences of 0's and 1's of length n. We can therefore label the Hasse diagram of a lattice $(P(A), \subseteq)$ by such sequence of 0's and 1's.

For example, lattices of P(A) and P(B) of the last example can be labeled as below:



The lattice so obtained is named B_n. The properties of the partial order in B_n can be described directly as follows:

Let $x = a_1 a_2 ..... a_n$ and $y = b_1 b_2 ..... b_n$ be any two elements of B_n. Then

(1) $x \leq y$ if and only if $a_k < b_k$, k = 1, 2,.....,n, where a_k and b_k are 0 or 1.

(2) $x \wedge y = c_1 c_2 .... c_n$, where $c_k = \min (a_k, b_k)$

(3) $x \vee y = d_1\, d_2 \ldots d_n$, where $d_k = \max(a_k, b_k)$

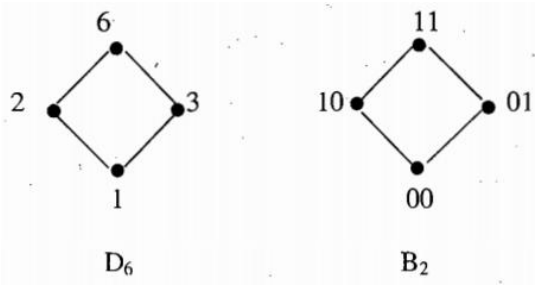
(4) x has a complement $x' = z_1\, z_2 \ldots\ldots z_n$ where $z_k = 1$ if $x_k = 0$ and $z_k = 0$ if $x_k = 1$.

Remark: $(B_n, \leq)$ under the partial order $\leq$ defined above is isomorphic to $(P(A), \subseteq)$, when A has n elements. In such a case $x \leq y$ corresponds to $S \subseteq T$, $x \vee y$ corresponds to $S \cup T$ and x' corresponds to A^c.

Example: Let $D_6 = \{1, 2, 3, 6\}$, set of divisors of 6. Then D_6 is isomorphic to B_2. In fact, $f : D_6 \to B_2$ defined by

$$f(1) = 00,\ f(2) = 10,\ f(3) = 01,\ f(6) = 11$$

is an isomorphism.



Example: Let $A = \{a, b\}$ and $P(A) = \{\varphi, \{a\}, \{b\}, \{a, b\}\}$. Then the lattice $(P(A), \subseteq)$ is isomorphic to the lattice $(D_6, |)$ with divisibility as the partial order relation.
In fact, we define a mapping $f : D_6 \to P(A)$ by

$$f(1) = \varphi,\ f(2) = \{a\},\ f(3) = \{b\},\ f(6) = \{a, b\}$$

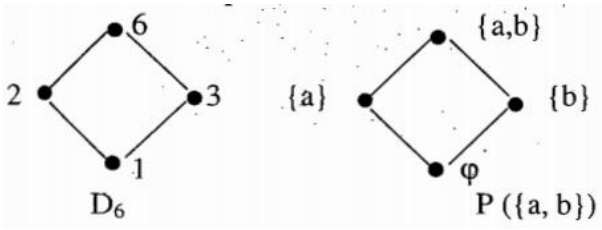
Then f is bijective and we see that

$$1|2 \Leftrightarrow \{\varphi\} \subseteq \{a\} \Leftrightarrow f(1) \subseteq f(2)$$

$$2|6 \Leftrightarrow \{a\} \subseteq \{a, b\} \Leftrightarrow f(2) \subseteq f(6)$$

and so on.

Hence f is isomorphism.



3.2.3 Lattice Endomorphism

Definition: Let $(L, \vee, \wedge)$ be a lattice. Then lattice homomorphism $f : L \to L$ is called an **endomorphism**.

If $f : L \to L$ is an endomorphism, then the image set of f is sublattice of L.

3.2.4 Lattice Automorphism

Definition: Let $(L, \vee, \wedge)$ be lattice. Then the lattice isomorphism $f : L \to L$ is called an **automorphism**.

3.3 Complete Lattices

Let $(L, \vee, \wedge)$ be a lattice and let $S = \{a_1, a_2, ..., a_n\}$ be a finite subset of L. Then

LUB of S is represented by $a_1 \vee a_2 \vee .....\vee a_n$

GLB of S is represented by $a_1 \wedge a_2 \wedge \ldots\ldots\wedge a_n$

Definition: A lattice is called **complete** if each of its non-empty subsets has a least upper bound and a greatest lower bound.

Obviously, every finite lattice is complete.

Also every complete lattice must have a least element, denoted by 0 and a greatest element, denoted by I. The least and greatest elements if exist are called **bound (units, universal bounds)** of the lattice.

3.4 Bounded Lattices

Definition: A lattice L is said to be **bounded** if it has a greatest element I and a least element 0.

For the lattice $(L, \vee, \wedge)$ with $L = \{a_1, a_2, ..., a_n\}$,

$$a_1 \vee a_2 \vee \ldots\ldots\ldots \vee a_n = I$$

and $a_1 \wedge a_2 \wedge \ldots\ldots\ldots \wedge a_n = 0$

Example: The lattice $\mathbf{Z}^+$ of all positive integers under partial order of divisibility is not a bounded lattice since it has a least element (the integer 1) but no greatest element.

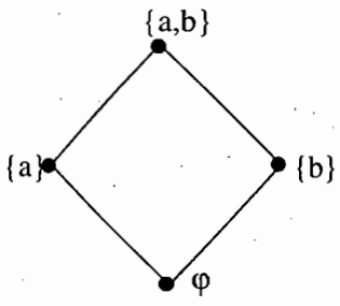
Example: The lattice **Z** of integers under partial order $\leq$ (less than or equal to) is not bounded since it has neither a greatest element nor a least element.

Example: Let A be a non-empty set. Then the lattice $(P(A), \subseteq)$ is bounded. Its greatest element is A and the least element is empty set φ.

In particular, if A has two elements, say a and b.

Then $P(A) = \{\varphi, \{a\}, \{b\}, \{a, b\}\}$.

The Hasse diagram of $(P(A), \subseteq)$ is shown below :



Here greatest element of the lattice $(P(A), \subseteq)$ is $\{a, b\}$ and least element is empty set φ. So the lattice $(P(A), \subseteq)$ is bounded, where $A = \{a, b\}$.

In general, if $(L, \leq)$ is a bounded Lattice, then for all $a \in L$

$$0 \leq a \leq I$$

$a \vee 0 = a, \quad a \wedge 0 = 0$

$a \vee I = I, \quad a \wedge I = a$

Thus 0 acts as identity of the operation $\vee$ and I acts as identity of the operation $\wedge$.

3.5 Complemented Lattices

3.5.1 Definition: Let $(L \vee, \wedge, 0, I)$ be a bounded lattice with greatest element I and the least element 0. Let $a \in L$. Then an element $b \in L$ is called a **complement** of a if

$$a \vee b = I$$

and $$a \wedge b = 0$$

It follows from this definition that

0 and I are complement of each other.

Further, I is the only complement of 0. For this, suppose that $c \neq I$ is a complement of 0 and $c \in L$, then

$$0 \vee c = I \text{ and } 0 \wedge c = 0$$

But $0 \vee c = c$.

Therefore, we have $c = I$ which contradicts $c \neq I$.

Similarly, 0 is the only complement of I.

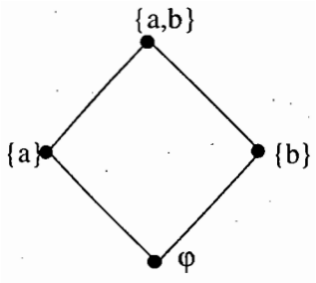
3.5.2 Definition: A lattice $(L, \vee, \wedge, 0, I)$ is called **complemented** if it is bounded and if every element of L has at least one complement.

Example: The lattice $(P(A), \subseteq)$ of the power set of any set A is a bounded lattice, where meet and join operations on P(A) are $\cap$ and $\cup$ respectively. Its bounds are φ and A. The lattice $(P(A), \subseteq)$ is complemented in which the complement of any subset B of A is A – B.

In particular, if A has two elements, say a and b.

Then $P(A) = \{\varphi, \{a\}, \{b\}, \{a, b\}\}$.

The Hasse diagram of $(P(A), \subseteq)$ is then as shown below :



Here complement of φ is {a, b}, i.e. φ′ = {a, b}

complement of {a, b} is φ, i.e. {a, b}′ = φ

complement of {a} is {b}, i.e. {a}′ = {b}

complement of {b} is {a}, i.e. {b}′ = {a}

Thus, the lattice (P(A), ⊆) is complemented, where A = {a, b}.

3.6 Distributive and Non- distributive Lattices

3.6.1 Definition: A lattice (L, ∨, ∧) is called a **distributive lattice** if for any elements a, b and c in L,

(l) $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

(2) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

Properties (1) and (2) are called **distributive properties.**

Thus, in a distributive lattice, the operations ∧ and ∨ are distributive over each other.

We further note that, by the principle of duality, the condition (1) holds if and only if (2) holds. Therefore it is sufficient to verify any one of these two equalities for all possible combinations of the elements of a lattice.

If a lattice L is not distributive, we say that L is **non-distributive**.

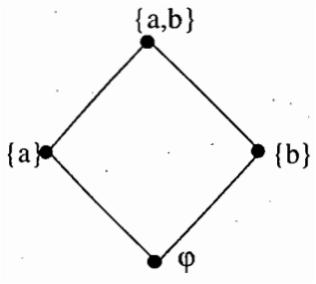
Example: For a set S, the lattice (P(S), ⊆) is distributive. The meet and join operations in P(S) are ∩ and ∪ respectively. Also we know, by set theory, that for A, B, C ∈ P(S),

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

and $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

In particular, if S has two elements, say a and b.

Then P(S) = {φ, {a}, {b}, {a, b}}. The Hasse diagram of (P(S), ⊆) is then as shown below:



Then we have

$\varphi \cap (\{a\} \cup \{b\}) = \varphi \cap \{a, b\} = \varphi$

Also $(\varphi \cap \{a\}) \cup (\varphi \cap \{b\}) = \varphi \cup \varphi = \varphi$

Thus we have

$\varphi \cap (\{a\} \cup \{b\}) = (\varphi \cap \{a\}) \cup (\varphi \cap \{b\})$

Similarly,

$\varphi \cup (\{a\} \cap \{b\}) = (\varphi \cup \{a\}) \cap (\varphi \cup \{b\})$

Thus distributive laws are satisfied for the elements φ, {a} and {b} of P(S), where S = {a, b}.

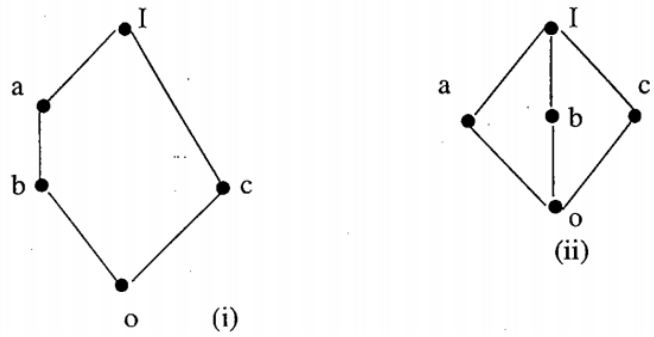
So for any A, B, C ∈ P(S) where S = {a, b}, we have

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

and $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Thus, the lattice (P(S), ⊆) is distributive, where S = {a, b}.

Example: The five elements lattices given in the following diagrams are non- distributive.



In fact for the lattice (i), we see that

$$a \wedge (b \vee c) = a \wedge I = a,$$

while

$$(a \wedge b) \vee (a \wedge c) = b \vee 0 = b$$

Hence

$$a \wedge (b \vee c) \neq (a \wedge b) \vee (a \wedge c)$$

This shows that (i) is non- distributive.

For the lattice (ii), we have

$$a \wedge (b \vee c) = a \wedge I = a,$$

while

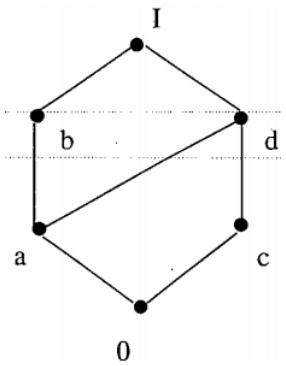
$$(a \wedge b) \vee (a \wedge c) = 0 \vee 0 = 0$$

Hence

$$a \wedge (b \vee c) \neq (a \wedge b) \vee (a \wedge c) ,$$

This shows that (ii) is also non- distributive.

Example: The lattice shown in the diagram below is distributive:



The distributive properties are satisfied for any ordered triplet chosen from the given elements.

3.6.2 Theorem: The direct product of any two distributive lattices is a distributive lattice.

Proof: Let $(L_1, \leq_1)$ and $(L_2, \leq_2)$ be two lattices in which meet and join are $\wedge_1$, $\vee_1$ and $\wedge_2$, $\vee_2$ respectively. Then meet and join in $L_1 \times L_2$ are defined by

$$(a_1, b_1) \wedge (a_2, b_2) = (a_1 \wedge_1 a_2, b_1 \wedge_2 b_2) \quad (1)$$

and

$$(a_1, b_1) \vee (a_2, b_2) = (a_1 \vee_1 a_2, b_1 \vee_2 b_2) \quad (2)$$

Since L_1 is distributive, we have

$$a_1 \wedge_1 (a_2 \vee_1 a_3) = (a_1 \wedge_1 a_2) \vee_1 (a_1 \wedge_1 a_3) \quad (3)$$

Since L_2 is distributive, we have

$$b_1 \wedge_2 (b_2 \vee_2 b_3) = (b_1 \wedge_2 b_2) \vee_2 (b_1 \wedge_2 b_3) \quad (4)$$

Therefore,

$(a_1, b_1) \wedge [(a_2, b_2) \vee (a_3, b_3)]$

$$= (a_1, b_1) \wedge [(a_2 \vee_1 a_3, b_2 \vee_2 b_3)]$$

$$= (a_1 \wedge_1 (a_2 \vee_1 a_3), b_1 \wedge_2 (b_2 \vee_2 b_3))$$

$$= ((a_1 \wedge_1 a_2) \vee_1 (a_1 \wedge_1 a_3), (b_1 \wedge_2 b_2) \vee_2 (b_1 \wedge_2 b_3)) \quad (5)$$

[Using (3) and (4)]

Now we have

$[(a_1, b_1) \wedge (a_2, b_2)] \vee [(a_1, b_1) \wedge (a_3, b_3)]$

$= (a_1 \wedge_1 a_2, b_1 \wedge_2 b_2) \vee (a_1 \wedge_1 a_3, b_1 \wedge_2 b_3)$ [using (1)]

$= ((a_1 \wedge_1 a_2) \vee_1 (a_1 \wedge_1 a_3), (b_1 \wedge_2 b_2) \vee_2 (b_1 \wedge_2 b_3))$ [using (2)] (6)

Hence from (5) and (6), we obtain

$(a_1, b_1) \wedge [(a_2, b_2) \vee (a_3, b_3)] = [(a_1, b_1) \wedge (a_2, b_2)] \vee [(a_1, b_1) \wedge (a_3, b_3)]$

This proves that $L_1 \times L_2$ is distributive.

3.7 Join- irreducible Elements in Lattices

Definition: Let $(L, \wedge, \vee)$ be a lattice. An element $a \in L$ is said to be **join- irreducible** if it cannot be expressed as the join of two distinct elements of L.

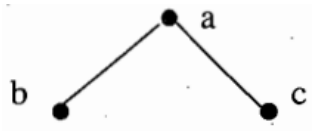
In other words, $a \in L$ is join- irreducible if for any $b, c \in L$

$a = b \vee c \Rightarrow a = b$ or $a = c$

For example, prime numbers under multiplication have this property. In fact, if p is a prime number, then $p = a\,b \Rightarrow p = a$ or $p = b$.

Clearly 0 is join- irreducible.

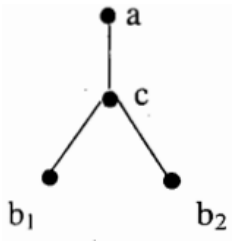
Further, suppose that 'a' has at least two immediate predecessors, say b and c as in the diagram below:



Then $a = b \vee c$ and so **a is not join - irreducible.**

On the other hand if 'a' has a unique immediate predecessor 'c', then

$a \neq \sup(b_1, b_2) = b_1 \vee b_2$ for any other elements b_1 and b_2 because c would lie between b_1, b_2 and a.

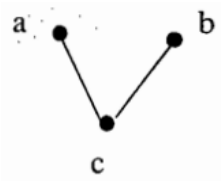


In other words, **$a \neq 0$ is join- irreducible if and only if 'a' has a unique predecessor.**

3.8 Atoms in Lattices

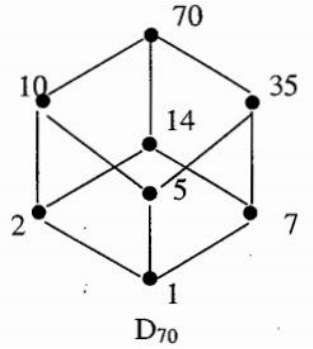
Definition: Those elements, which immediately succeed 0, are called **atoms**.

From the above discussion, it follows that the **atoms are join- irreducible.**



Example: Consider the lattice

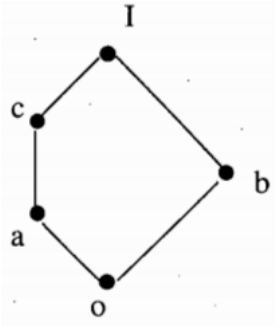
$D_{70} = \{1, 2, 5, 7, 10, 14, 35, 70\}$



Then the set of atoms of D_{70} is

$$A = \{2, 5, 7\}$$

However, lattices can have other join- irreducible elements. For example, the element 'c' in five-element lattice (shown below in figure) is not an atom, even then it is join- irreducible because it has only one immediate predecessor, namely a.



Let 'a' be an element of a finite lattice which is not join-irreducible, then we can write

$$a = b \vee c$$

If b and c are not join- irreducible, then we can write them as the join of other elements. Since L is finite, we shall finally have

$$a = d_l \vee d_2 \vee d_3 \vee ..... \vee d_n , \qquad (1)$$

where $d_i = l, 2, ..., n$ are join- irreducible.

If d_i precedes d_j, then $d_i \vee d_j = d_j$, so we delete d_i from the expression. Thus d's are irredundant, i.e., no d precedes any other d.

The expression (1) need not be unique.

For example, in the five- element lattice shown above, we have

$$I = a \vee b \quad \text{and} \quad I = b \vee c$$

Remarks:

I. Let $(L, \wedge, \vee)$ be a finite distributive lattice. Then every element 'a' in L can be written uniquely (except for order) as the join of irredundant join- irreducible elements.

II. Let L be a complemented lattice with unique complements. Then the join- irreducible elements of L, other than 0, are its atoms.

III. Let L be a finite complemented distributive lattice. Then every element 'a' in L is the join of a unique set of atoms.

3.9 Check Your Progress

1. Let A = {a, b} and P(A) = {ϕ, {a}, {a, b}} then the lattice $(P(A), \subseteq)$ is isomorphic to the lattice (D6, 1) with divisibility as the partial order relation.

2. Consider the lattice $(D_{70}, |)$, where

D_{70} = {1, 2, 5, 7, 10, 14, 35, 70}

Then the set of atoms of D_{70} is

A = {2, 5, 7}

The unique representation of each non-atom by atoms is

$10 = 2 \vee 5$

$14 = 2 \vee 7$

$35 = 5 \vee 7$

$2 \vee 5 \vee 7$

3. Let L be a bounded distributive lattice. If a complement of any element exists, then it is unique.

4. Every chain is a distributive lattice.

Sol. Let $(L, \leq)$ be a chain and $a, b, c \in L$. We shall show that distributive law holds for any $a, b, c \in L$.

Two cases arise:

Case 1. Let $a \leq b$ or $a \leq c$

In this case, we have

$$a \wedge (b \vee c) = a \qquad (1)$$

and

$$(a \wedge b) \vee (a \wedge c) = a \qquad (2)$$

Hence from (1) and (2), we obtain

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

Also, by Principle of Duality, we have

$$a \wedge (b \vee c) = (a \vee b) \wedge (a \vee c)$$

Case II. Let $b \leq a$ or $c \leq a$

Then we have

$$a \wedge (b \vee c) = (b \vee c) \qquad (3)$$

and

$$(a \wedge b) \vee (a \wedge c) = (b \vee c) \qquad (4)$$

Hence from (3) and (4), we obtain

$$a \wedge (b \vee c) = (b \vee c)$$

Hence distributive law holds for any $a, b, c \in L$.

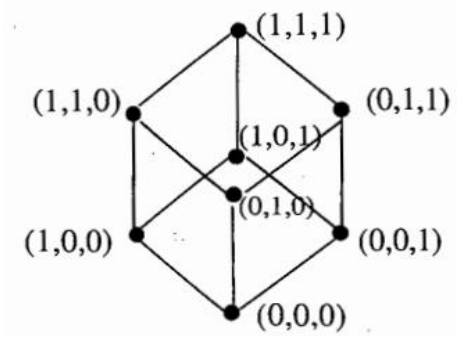
5. Let L^n be the lattice of n tuples of 0 and 1, where partial ordering is defined for $a = (a_1, a_2,...,a_n)$, $b = (b_1, b_2, ..., b_n) \in L^n$ by

$$a \leq_n b \Leftrightarrow a_i \leq b_i \qquad \text{for all } i = 1, 2, ..., n ,$$

where $\leq$ means less than or equal to.

Then $(L^n, \leq_n)$ is lattice which is bounded.

For example, the bounds are (0, 0, 0) and (1, 1, 1) for L^3.



The complement of an element of L^n can be obtained by interchanging 1 by 0 and 0 by 1 in the n- tuple representing the element. For example, complement of (1, 0, 1) in L^3 is (0, 1, 0).

3.10 Summary

In this chapter, we have discussed about lattice isomorphism, some special lattices namely, complete lattice, bounded lattice, complemented lattice and distributive lattice. Examples are also given related to these special lattices. We have also studied join- irreducible elements and atoms in a lattice.

3.11 Keywords

Lattice isomorphism, Bounded lattice, complemented lattice, distributive lattice, join-irreducible elements, atoms

3.12 Self-Assessment Test

1. Show that Every finite lattice *L* is bounded.

2. Prove that if *L* be a finite distributive lattice. Then every *a* in *L* can be written uniquely (except for order) as the join of irredundant join irreducible elements.

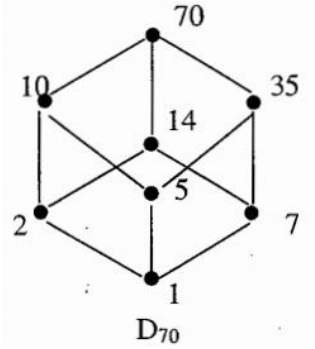
3. Let *L* be a bounded distributive lattice. Then complements are unique if they exist.

4. Let *L* be a finite complemented distributive lattice. Then every element *a* in *L* is the join of a unique set of atoms.

3.13 Answers to check your progress

1. In fact, we define a mapping f : D6 → P(A) by f(1) = ϕ, f(2) = {a}, f(3) = {b}, f(6) = {a, b} , then f is bijective and we note that 1|2 ⇔ {ϕ} ⊆ {a} ⇔ f(1) ⊆ f(2) 2|6 ⇔ {a} ⊆ {a, b} ⇔ f(2) ⊆ f(6) and so on. Hence f is isomorphism.

2.



D_{70}

3. Suppose on the contrary that b and c are complements of the element $a \in L$. Then

$a \vee b = I \quad , \quad a \wedge b = 0 \qquad (1)$

and $a \vee c = I \quad , \quad a \wedge c = 0 \qquad (2)$

Now we have

$b = b \vee 0$

$= b \vee (a \wedge c)$ [using (2)]

$= (b \vee a) \wedge (b \vee c)$ [Using distributive law]

$= (a \vee b) \wedge (b \vee c)$ [Using commutative law]

$= I \wedge (b \vee c)$ [using (1)]

$= b \vee c$

$\Rightarrow \quad b = b \vee c \qquad (3)$

Similarly,

$c = c \vee 0$

$= c \vee (a \wedge b)$ [using (1)]

$= (c \vee a) \wedge (c \vee b)$ [Using distributive law]

$= (a \vee c) \wedge (c \vee b)$ [Using commutative law]

$= I \wedge (c \vee b)$ [using (2)]

$= I \wedge (b \vee c)$

$= b \vee c$

$\Rightarrow \quad c = b \vee c$ (4)

Then from (3) and (4), we obtain b = c. This contradicts our supposition. So our supposition is wrong. Hence if a complement of any element exists in a bounded distributive lattice, then it is unique.

3.14 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.

2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.

3 C.L. Liu, Elements of Discrete Mathematics, McGraw-Hili Book Co.

4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 4 **Written by Dr. Vizender Singh**

BOOLEAN ALGEBRA - I

Structure:

4.0 Learning Objectives

Objective of this chapter is to study algebraic structure like Boolean algebra, which has much application in computers.

4.1 Introduction

In the previous chapters we have studied about basic definition and concepts of lattices. In the present chapter we will define one more algebraic structure namely, Boolean algebra**.** Boolean algebra is a significant tool for the analysis and design of electronic computers. It has wide applications to switching theory and logical design of electronic circuits. We will study about basic definition of Boolean algebra, various Boolean identities, sub- Boolean algebra, direct product of Boolean algebras and Boolean homomorphism. Examples are also given to illustrate these topics.

4.2 Definitions of Boolean Algebra and Related Examples

4.2.1 Definition: A non- empty set B with two binary operations $\vee$ and $\wedge$, a unary operation $'$, and two distinct elements 0 and I is called a **Boolean Algebra** if the following axioms holds for any elements a, b, c $\in$ B:

[B_1]: Commutative Laws:

$a \vee b = b \vee a$ and $a \wedge b = b \wedge a$

[B_2]: Distributive Laws:

$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

[B_3]: Identity Laws:

$a \vee 0 = a$ and $a \wedge I = a$

[B_4]: Complement Laws:

$a \vee a' = I$ and $a \wedge a' = 0$

We shall call 0 as zero element, I as unit element and a' the complement of a.

We denote a Boolean algebra by $(B, \vee, \wedge, ', 0, I)$.

Example 1: Let A be a non-empty set and P(A) be its power set. Then the set algebra $(P(A), \cup, \cap, -, \phi, A)$ is a Boolean algebra.

Example 2: Let $B = \{0, 1\}$ be the set of bits (binary digits) with the binary operations $\vee$ and $\wedge$ and the unary operation $'$ defined by the following tables:

$\vee$	1	0
1	1	1
0	1	0

,

$\wedge$	1	0
1	1	0
0	0	0

$'$	1	0
	0	1

Here the operations $\vee$ and $\wedge$ are logical operations and complement of 1 is 0 whereas complement of 0 is 1. Then $(B, \vee, \wedge, ', 0, 1)$ is a Boolean Algebra. It is the simplest example of a two- element algebra.

Further, a two element Boolean algebra is the only Boolean algebra whose diagram is a chain.

This algebra is known as **Switching Algebra** and represents a switching network with n inputs and one output.

Example 4: Let S be the set of statement formulas involving n statement variables. The algebraic system $(S, \wedge, \vee, \sim, F, T)$ is a Boolean algebra in which $\wedge$, $\vee$, $\sim$ denotes the operations of conjunction, disjunction and negation respectively. The element F and T denotes the formulas which are contradictions and tautologies respectively. The partial ordering corresponding to $\wedge$, $\vee$ is implication $\Rightarrow$.

We have seen that B_n is a Boolean algebra. Using this fact, we can also define Boolean algebra as follows:

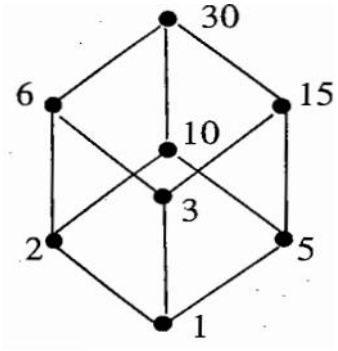
4.2.2 Definition: A finite lattice is called a **Boolean algebra** if it is isomorphic with B_n for some non- negative integer n.

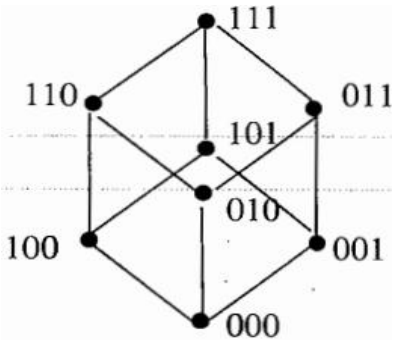
For example, D_{30} is isomorphic to B_3. In fact, the mapping $f : D_{30} \to B_3$ defined by

$f(1) = 000$, $f(2) = 100$, $f(3) = 010$, $f(5) = 001$,

f(6) = 110, f(10) = 101, f(15) = 011, f(30) = 111

is an isomorphism. Hence D_{30} is a Boolean algebra.

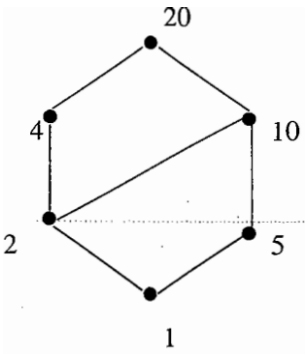




Remarks:

1. If a finite lattice L does not contain 2^n elements for some non- negative integer n, then L cannot be a Boolean algebra.

For example, consider D_{20} = {1, 2, 4, 5, 10, 20} that has 6 elements and $6 \neq 2^n$ for any integer $n \geq 0$. Therefore, D_{20} is not a Boolean algebra.



2. If | L | = 2^n, then L may or not be a Boolean algebra. If L is isomorphic to B_n, then it is Boolean algebra, otherwise it is not.

For example, consider D_{20}, D_{30}, D_{210}, D_{66}, D_{646}. We notice that

(i) 20 cannot be represented as product of distinct primes and so D_{20} is not a Boolean algebra.

(ii) 30 = 2.3.5, where 2, 3, 5 are distinct primes. Hence D_{30} is a Boolean algebra.

(iii) 210 = 2.3.5.7 (all distinct primes) and so D_{210} is a Boolean algebra.

(iv) 66 = 2.3.11 (product of distinct primes) and so D_{66} is a Boolean algebra.

(v) 646 = 2.17.19 (product of distinct primes) and so D_{646} is a Boolean algebra.

4.2.3 Duality: The **dual of any statement** in a Boolean algebra B is obtained by interchanging $\vee$ and $\wedge$ and interchanging the zero element and unit element in the original statement.

For example, the dual of $a \wedge 0 = 0$ is $a \wedge I = I$.

Principle of duality: The dual of any theorem in a Boolean algebra is also a theorem.

Thus, dual theorem is proved by using the dual of each step of the proof of the original statement.

4.3 Properties of Boolean Algebra

4.3.1 Idempotent, Boundedness, Absorption and Associative Laws

Theorem: Let a, b and c be any elements in a Boolean algebra $(B, \vee, \wedge, ', 0, I)$. Then

1. Idempotent Laws:

(i) $a \vee a = a$ (ii) $a \wedge a = a$

2. Boundedness Laws:

(i) $a \vee I = I$ (ii) $a \wedge 0 = 0$

3. Absorption Laws:

(i) $a \vee (a \wedge b) = a$ (ii) $a \wedge (a \vee b) = a$

4. Associative Laws:

(i) $(a \vee b) \vee c = a \vee (b \vee c)$ (ii) $(a \wedge b) \wedge c = a \wedge (b \wedge c)$

Proof: It is sufficient to prove first part of each law since second part follows from the first by principle of duality.

1(i): We have

$a = a \vee 0$ (by identity law)

$= a \vee (a \wedge a')$ (by complement law)

$= (a \vee a) \wedge (a \vee a')$ (by distributive law)

$= (a \vee a) \wedge I$ (by complement law)

$= a \vee a$ (by identity law)

Hence $a \vee a = a$

which proves 1(i).

2(i): We have

$a \vee I = (a \vee I) \wedge I$ (by identity law)

$= (a \vee I) \wedge (a \vee a')$ (by complement law)

$= a \vee (I \wedge a')$ (by Distributive law)

$= a \vee a'$ (by identity law)

$= I$ (by complement law)

Hence $a \vee I = I$

3(i): We have

$a \vee (a \wedge b) = (a \wedge I) \vee (a \wedge b)$ (by identity law)

$= a \wedge (I \vee b)$ (by distributive law)

$= a \wedge (b \vee I)$ (by commutative law)

$= a \wedge I$ (by Identity law)

$= a$ (by identity law)

4(i) Let

$L = (a \vee b) \vee c$ and $R = a \vee (b \vee c)$

Then

$a \wedge L = a \wedge [(a \vee b) \vee c]$

$= [a \wedge (a \vee b)] \vee (a \wedge c)$ (by distributive Law)

$= a \vee (a \wedge c)$ (by absorption law)

$= a$ (by absorption law)

and

$a \wedge R = a \wedge [a \vee (b \vee c)]$

$= (a \wedge a) \vee (a \wedge (b \vee c)]$ (by distributive law)

$= a \vee (a \wedge (b \vee c)]$ (by idempotent law)

$= a$ (by absorption Law)

Thus $a \wedge L = a \wedge R$ and so, by duality, $a \vee L = a \vee R$.

Further,

$a' \wedge L = a' \wedge [(a \vee b) \vee c]$

$= [a' \wedge (a \vee b)] \vee (a' \wedge c)$ (by distributive law)

$= [(a' \wedge a) \vee (a' \wedge b)] \vee (a' \wedge c)$ (by distributive law)

$= [0 \vee (a' \wedge b)] \vee (a' \wedge c)$ (by complement Law)

$= (a' \wedge b) \vee (a' \wedge c)$ (by Identity law)

$= a' \wedge (b \vee c)$ (by distributive law)

$\therefore \quad a' \wedge L = a' \wedge (b \vee c)$

On the other hand,

$a' \wedge R = a' \wedge [a \vee (b \vee c)]$

$= (a' \wedge a) \vee [a' \wedge (b \vee c)]$ (by distributive law)

$= 0 \vee [a' \wedge (b \vee c)]$ (by complement law)

$= a' \wedge (b \vee c)$ (by identity law)

$\therefore \quad a' \wedge R = a' \wedge (b \vee c)$

Hence

$a' \wedge L = a' \wedge R$ and so by duality $a' \vee L = a' \vee R$

Now we have

$L = 0 \vee L$ (by identity law)

$= (a \wedge a') \vee L$ (by complement law)

$= (a \vee L) \wedge (a' \vee L)$ (by distributive law)

$= (a \vee R) \wedge (a' \vee R)$ (using $a \vee L = a \vee R$ and $a' \vee L = a' \vee R$)

$= (a \wedge a') \vee R$ (by distributive law)

$= 0 \vee R$ (by complement law)

$= R$ (by identity law)

Therefore, we have

$L = R$

Hence

$(a \vee b) \vee c = a \vee (b \vee c)$,

which completes the proof of the theorem.

4.3.2 Uniqueness of Complement and Involution Law

Theorem: Let 'a' be any element of a Boolean algebra B. Then

(i) Complement of 'a' is unique **(uniqueness of complement)**

(ii) $(a')' = a$ **(Involution law)**

(iii) $0' = I$ and $I' = 0$

Proof: (i) Let a' and x be two complements of $a \in B$. Then

$a \vee a' = I$ and $a \wedge a' = 0$ (1)

Also $a \vee x = I$ and $a \wedge x = 0$ (2)

Now we have

$a' = a' \vee 0$ (Identity law)

Therefore, $a' = a' \vee (a \wedge x)$ [by (2)]

$= (a' \vee a) \wedge (a' \vee x)$ (Distributive law)

$= I \wedge (a' \vee x)$ [by (1)]

$= a' \vee x$ [Identity law]

So, we have $a' = a' \vee x$ (3)

Also we have

$x = x \vee 0$ (Identity law)

$= x \vee (a \wedge a')$ [by (1)]

$= (x \vee a) \wedge (x \vee a')$ (Distributive law)

$= I \wedge (x \vee a')$ [by (2)]

$= x \vee a'$ (Identity law)

$= a' \vee x$ (commutative law)

So, we have $x = a' \vee x$ (4)

Hence from (3) and (4), we obtain $a' = x$ and so complement of any element in B is unique.

(ii) Let a' be a complement of a. Then

$a \vee a' = I$ and $a \wedge a' = 0$

By commutativity, the above relations become

$a' \vee a = I$ and $a' \wedge a = 0$

This implies that a is complement of a', that is, $a = (a')'$

(iii) By boundedness law, we have

$0 \vee I = I$

Also by identity law, we have

$0 \wedge I = 0$

These two relations imply that I is the complement of 0, i.e. $I = 0'$

By principle of duality, we have then

$0 = I'$

4.3.3 De - Morgan's Laws in a Boolean Algebra

Theorem: Let a, b be elements of a Boolean algebra. Then

$(a \vee b)' = a' \wedge b'$ and $(a \wedge b)' = a' \vee b'$.

Proof: We have

$(a \vee b) \vee (a' \wedge b') = (b \vee a) \vee (a' \wedge b')$ (commutative law)

$= b \vee (a \vee (a' \wedge b'))$ (associative law)

$= b \vee [(a \vee a') \wedge (a \vee b')]$ (distributive law)

$= b \vee [\, I \wedge (a \vee b')]$ (complement law)

$= b \vee (a \vee b')$ (identity law)

$= b \vee (b' \vee a)$ (commutative law)

$= (b \vee b') \vee a$ (associative law)

$= I \vee a$ (complement law)

$= I$ (Boundedness law)

So we have

$(a \vee b) \vee (a' \wedge b') = I$ (1)

Also

$(a \vee b) \wedge (a' \wedge b') = [(a \vee b) \wedge a'\,] \wedge b'$ (associative law)

$= [(a \wedge a') \vee (b \wedge a')] \wedge b'$ (distributive law)

$= [0 \vee (b \wedge a')] \wedge b'$ (complement law)

$= (b \wedge a') \wedge b'$ (identity law)

$= (b \wedge b') \wedge a'$ (associative law)

$= 0 \wedge a'$ (complement law)

$= 0$ (Boundedness law)

So we have

$$(a \vee b) \wedge (a' \wedge b') = 0 \qquad (2)$$

Hence from (1) and (2), we obtain $a' \wedge b'$ is complement of $a \vee b$, i.e. $(a \vee b)' = a' \wedge b'$.

The second part follows by principle of duality.

4.4 Boolean Algebra in terms of Lattice

We have proved already that Boolean algebra $(B, \vee, \wedge, ', 0, I)$ satisfies associative laws, commutative law and absorption law. Hence every Boolean algebra is a lattice with join as $\vee$ and meet as $\wedge$. Also boundedness laws hold in a Boolean algebra. Thus Boolean algebra becomes a bounded lattice. Also Boolean algebra obeys distributive law and is complemented. Conversely, every bounded, distributive and complemented lattice satisfies all the axioms of a Boolean algebra. Hence we can define a Boolean algebra as

Definition: A Boolean algebra is a **bounded distributive and complemented lattice.**

Now, being a lattice, a Boolean algebra must have a partial ordering. Recall that in case of lattice, we had defined partial ordering $\leq$ by $a \leq b$ if $a \vee b = b$ or $a \wedge b = a$.

The following result yields much more than these required conditions:

4.4.1 Theorem: If a, b are in a Boolean algebra, then the following are equivalent:

(1) $a \vee b = b$

(2) $a \wedge b = a$

(3) $a' \vee b = I$

(4) $a \wedge b' = 0$

Proof: (1) $\Leftrightarrow$ (2), proved already in theorem 2.5.1.

For (1) $\Rightarrow$ (3): Suppose $a \vee b = b$, then

$a' \vee b = a' \vee (a \vee b)$

$= (a' \vee a) \vee b$ (associative law)

$= I \vee b$ (complement law)

$= I$ (boundedness law)

So we have if $a \vee b = b$, then $a' \vee b = I$

i.e. (1) $\Rightarrow$ (3)

Conversely, suppose $a' \vee b = I$, then

$a \vee b = I \wedge (a \vee b)$ (identity law)

$= (a' \vee b) \wedge (a \vee b)$ (by assumption of (3))

$= (a' \wedge a) \vee b$ (distributive law)

$= 0 \vee b$ (complement law)

$= b$ (identity law)

So we have if $a' \vee b = I$, then $a \vee b = b$

i.e. (3) $\Rightarrow$ (1)

Thus we have (1) $\Leftrightarrow$ (3).

Now we show that (3) $\Leftrightarrow$ (4).

Suppose first that (3) holds. Then, we have

$0 = I'$ (complement law)

$= (a' \vee b)'$ (by assumption of (3))

$= a'' \wedge b'$ (De- Morgan's law)

$= a \wedge b'$ (Involution law)

So we have if $a' \vee b = I$, then $a \wedge b' = 0$

i.e. (3) $\Rightarrow$ (4)

Conversely, if (4) holds, then

$I = 0'$ (complement law)

$= (a \wedge b')'$ (by assumption of (4))

$= a' \vee b''$ (De- Morgan's law)

$= a' \vee b$ (Involution law)

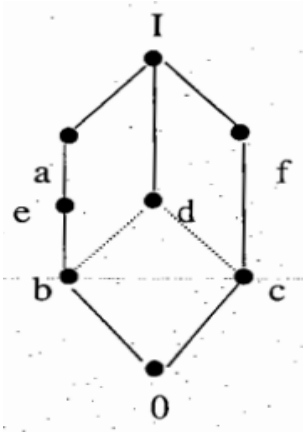
So we have if $a \wedge b' = 0$, then $a' \vee b = I$

i.e. (4) $\Rightarrow$ (3)

Thus we have (3) $\Leftrightarrow$ (4)

Hence all the four conditions are equivalent.

Example: Show that the lattice whose diagram is



is not a Boolean algebra.

Solution: Elements a and e are both complements of c since $c \vee a = I$, $c \wedge a = 0$ and $c \vee e = I$, $c \wedge e = 0$.

But in a Boolean algebra, complement of an element is unique. Hence the given lattice is not a Boolean algebra.

4.5 Sub- Boolean Algebra

Definition: Let $(B, \vee, \wedge, ', 0, I)$ be a Boolean algebra and $S \subseteq B$. If S contains the elements 0 and I and is closed under the operations $\vee$, $\wedge$ and $'$, then $(S, \wedge, \vee, ', 0, I)$ is called **Sub- Boolean Algebra.**

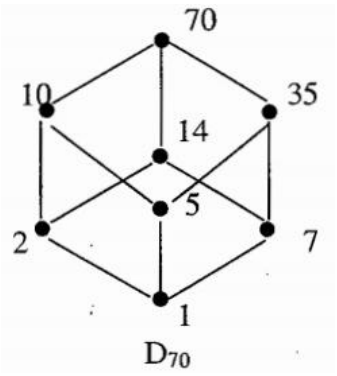
In practice, it is sufficient to check closure with respect to the set of operations $(\wedge, ')$ or $(\vee, ')$ for proving a subset S of B as the sub- Boolean algebra.

The definition of sub- Boolean algebra implies that it is a Boolean algebra.

A subset of Boolean algebra can be a Boolean algebra, but not necessarily a Boolean subalgebra because it is not closed with respect to the operations in B. For any Boolean algebra $(B, \wedge, \vee, ', 0, I)$, the subsets $\{0, I\}$ and the set B are both sub-Boolean algebras.

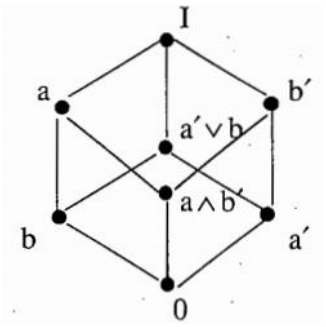
In addition to these sub-Boolean algebras, consider now any element $a \in B$ such that $a \neq 0$ and $a \neq I$ and consider the set $\{a, a', 0, I\}$. Obviously this set is a sub- Boolean algebra of the given Boolean algebra. Every element of a Boolean algebra generates a sub- Boolean algebra. More generally, any subset of B generates a sub- Boolean algebra.

For example, $D_{70} = \{1, 2, 5, 7, 10, 14, 35, 70\}$ is a Boolean algebra and $\{1, 2, 35, 70\}$ is a sub-Boolean algebra of D_{70}.



D_{70}

Example: Consider the Boolean algebra given in the diagram below:



Verify whether the following subsets are Boolean algebras or not:

$S_1 = \{a, a', 0, I\}$

$S_2 = \{a' \vee b, a \wedge b', 0, I\}$

$S_3 = \{a \wedge b', b', a, I\}$

$S_4 = \{b', a \wedge b', a', 0\}$

$S_5 = \{a, b', 0, I\}$

Solution: The subsets S_1 and S_2 are sub-Boolean algebras. The subsets S_3 and S_4 are Boolean algebras but not sub-Boolean algebras of the given Boolean algebra. The subset S_5 is not even a Boolean algebra.

4.6 Direct Product of Boolean Algebras

Definition: Let $(B_1, \wedge_1, \vee_1, ', 0_1, I_1)$ and $(B_2, \wedge_2, \vee_2, '', 0_2, I_2)$ be two Boolean algebras. The **Direct Product** of the two Boolean algebras is defined to be a Boolean algebra, denoted by $(B_1 \times B_2, \wedge_3, \vee_3, ''', 0_3, I_3)$ in which the operations are defined for any (a_1, b_1) and $(a_2, b_2) \in B_1 \times B_2$ as

$$(a_1, b_1) \wedge_3 (a_2, b_2) = (a_1 \wedge_1 a_2, b_1 \wedge_2 b_2)$$

$$(a_1, b_1) \vee_3 (a_2, b_2) = (a_1 \vee_1 a_2, b_1 \vee_2 b_2)$$

$$(a_1, b_1)''' = (a_1', b_1'')$$

$0_3 = (0_1, 0_2)$ and $I_3 = (I_1, I_2)$

Thus, from a Boolean algebra B, we can generate $B^2 = B \times B$, $B^3 = B \times B \times B$ etc.

4.7 Boolean Homomorphism

Definition: Let $(B, \wedge, \vee, ', 0, I)$ and $(P, \cap, \cup, —, \alpha, \beta)$ be two Boolean Algebras. A mapping $f : B \to P$ is called a **Boolean Homomorphism** if all the operations of the Boolean Algebra are preserved, that is, for any $a, b \in B$

$f(a \wedge b) = f(a) \cap f(b)$

$f(a \vee b) = f(a) \cup f(b)$

$f(a') = \overline{f(a)}$

$f(0) = \alpha$

$f(I) = \beta$

The above definition of homomorphism can be simplified by asserting that $f : B \to P$ preserves either the operations $\wedge$ and $'$ or the operations $\vee$ and $'$.

We now consider a mapping $g : B \to P$ in which the operations $\wedge$ and $\vee$ are preserved. Thus, g is a lattice homomorphism. Naturally g preserves the order and hence it maps the bounds 0 and I into the least and the greatest element respectively of the image set $g(B) \subseteq P$. It is however, not necessary that $g(0) = \alpha$ and $g(I) = \beta$. The complements, if defined in terms of g(0) and g(I) in g(B), are preserved, and $(g(B), \cap, \cup, —, g(0), g(I))$ is a Boolean algebra. Note that $g : B \to P$ is not a Boolean homomorphism, although $g : B \to g(B)$ is a Boolean homomorphism.

In any case, for any mapping from a Boolean Algebra which preserves the operations $\wedge$ and $\vee$, the image set is a Boolean algebra.

A Boolean homomorphism is called **Boolean isomorphism** if it is bijective.

4.8 Representation Theorem

Let B be a finite Boolean algebra. We know that an element 'a' in B is called an **atom (or minterm)** if 'a' immediately succeed the least element 0. Let A be the set of atoms of B and let P(A) be the Boolean algebra of all subsets of the set A of atoms. Then (by remark III in article 3.8), each $x \neq 0$ in B can be expressed uniquely (except for order) as the join of atoms (i.e. elements of A). So, let

$$x = a_1 \vee a_2 \vee \ldots\ldots \vee a_n$$

Consider the function

$$f : B \to P(A)$$

defined by

$$f(x) = \{a_1, a_2, \ldots\ldots, a_n\}$$

for each $x = a_1 \vee a_2 \vee .... \vee a_n$.

Stone's Representation Theorem: Any Boolean Algebra is isomorphic to a power set algebra $(P(S), \cap, \cup, \sim, \phi, S)$ for some set S.

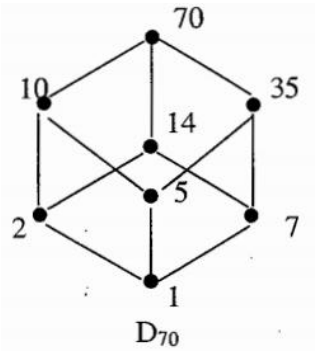
Restricting our discussion to finite Boolean Algebra B, the representation theorem can be stated as:

Theorem: Let B be a finite Boolean algebra and let A be the set of atoms of B. If P(A) is the Boolean Algebra of all subsets of the set A of atoms, then the mapping $f : B \to P(A)$ is an isomorphism.

Corollary: We know that if a set A has n elements, then its power set P(A) has 2^n elements. Thus, a finite Boolean algebra has 2^n elements for some positive integer n.

Example: Consider the Boolean algebra

$$D_{70} = \{1, 2, 5, 7, 10, 14, 35, 70\}$$



Then the set of atoms of D_{70} is

$A = \{2, 5, 7\}$

The unique representation of each non- atom by atoms is

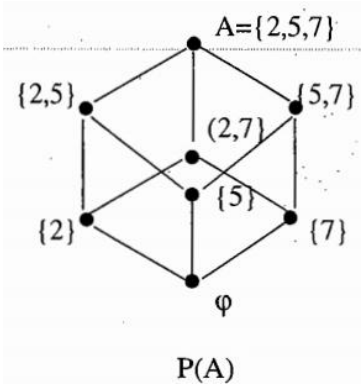
$10 = 2 \vee 5$

$14 = 2 \vee 7$

$35 = 5 \vee 7$

$70 = 2 \vee 5 \vee 7$

The diagram of the Boolean algebra of the power set P(A) of the set A of atoms is given below :



We observe that the diagram for D_{70} and P(A) are structurally the same.

4.9 Check Your Progress

1. The poset $D_{30} = \{1, 2, 3, 5, 6, 10, 15, 30\}$ has eight elements. Define $\vee$, $\wedge$ and $'$ on D_{30} by

$a \vee b = \text{lcm}(a, b)$, $a \wedge b = \gcd(a, b)$ and $a' = \frac{30}{a}$.

Then D_{30} is a Boolean algebra with 1 as the zero element and 30 as the unit element.

2. Let

$$n = p_1 p_2 \ldots\ldots p_k,$$

where p_i are distinct primes, known as set of atoms. Then D_n is a Boolean algebra.
(For large value of n, this theorem is used for determining whether D_n is a Boolean algebra or not.)

Proof: Let $A = \{p_1, p_2, \ldots, p_k\}$.
If $B \subseteq A$ and a_B is the product of primes in B, then $a_B \mid n$. Also any divisor of n must be of the form a_B for some subset B of A, where we assume that $a_\varphi = 1$. Further, if C and B are subsets of A, then $C \subseteq B$ if and only if $^{a}_{C} \mid ^{a}_{B}$.
Also

---

and

Thus the function $f: P(A) \to D_n$ defined by

$$f(B) = a_B$$

is an isomorphism. Since P(A) is a Boolean algebra, it follows that D_n is also a Boolean algebra.

3. Let B_n be the set of n tuples whose members are either 0 or 1.
Let $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$ be any two members of B_n. Then we define

$$a \vee_1 b = (a_1 \vee b_1, a_2 \vee b_2, \ldots, a_n \vee b_n)$$

and $$a \wedge_1 b = (a_1 \wedge b_1, a_2 \wedge b_2, \ldots, a_n \wedge b_n)$$

where $\vee$ and $\wedge$ are logical operations on {0, 1}, and

----------,

where ~ 0 = 1 and ~1 = 0 .

If 0_n represents (0, 0,.....,0) and 1_n = (1, 1,.....,1), then $(B_n, \vee_1, \wedge_1, ', 0_n, 1_n)$ is a Boolean algebra.

4. Let a, b and c be any elements in a Boolean algebra (B, $\vee$, $\wedge$, ′ , 0, I). Then show that (i) $a \vee I = I$ (ii) $a \wedge 0 = 0$.

5. Let a, b be elements of a Boolean algebra. Then $(a \vee b)' = a' \wedge b$.

4.10 Summary

In this chapter we have studied about one more algebraic structure namely, Boolean algebra**.** We have also discussed about various Boolean identities, sub- Boolean algebra, direct product of Boolean algebras and Boolean homomorphism. Examples were also given to illustrate these topics.

4.11 Keywords

Boolean algebra, sub- Boolean algebra, Boolean homomorphism

4.12 Self-Assessment Test

1. Write the dual of each Boolean equation:
(a) (a∗1) ∗ (0 ∗ a') = 0;
(b) a + a' b = a + b.

2. The set **D***m* of divisors of *m* is a bounded, distributive lattice with
a + b = a ∨ b = lcm(a, b) and a ∗ b = a ∧ b = gcd(a, b).
Show that **D***m* is a Boolean algebra if *m* is square free, i.e., if *m* is a product of distinct primes.

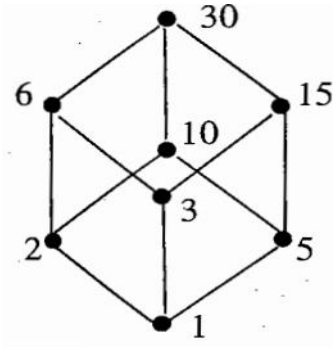
3. Consider the Boolean algebra **D**210.
(*a*) List its elements and draw its diagram.
(*b*) Find the set *A* of atoms.
(*c*) Find two subalgebras with eight elements.

4. Find the number of subalgebras of **D**210.

5. Find the atoms of $\mathbf{D}_m$.

4.13 Answers to check your progress

1.



2. (i) ${}^{a}_{C \cap B} = {}^{a}_{C} \wedge {}^{a}_{B} = \gcd({}^{a}_{C}, {}^{a}_{B})$

(ii) ${}^{a}_{C \cup B} = {}^{a}_{C} \vee {}^{a}_{B} = \text{lcm}({}^{a}_{C}, {}^{a}_{B})$

3. $a' = (\sim a_1, \sim a_2, \ldots, \sim a_n)$

4. We have

$a \vee I = (a \vee I) \wedge I$ (by identity law)

$= (a \vee I) \wedge (a \vee a')$ (by complement law)

$= a \vee (I \wedge a')$ (by Distributive law)

$= a \vee a'$ (by identity law)

$= I$ (by complement law)

Hence $a \vee I = I$.

5. We have

$(a \vee b) \vee (a' \wedge b') = (b \vee a) \vee (a' \wedge b')$ (commutative law)

$= b \vee (a \vee (a' \wedge b'))$ (associative law)

$= b \vee [(a \vee a') \wedge (a \vee b')]$ (distributive law)

$= b \vee [I \wedge (a \vee b')]$ (complement law)

$= b \vee (a \vee b')$ (identity law)

$= b \vee (b' \vee a)$ (commutative law)

$= (b \vee b') \vee a$ (associative law)

$= I \vee a$ (complement law)

$= I$ (Boundedness law)

So we have

$(a \vee b) \vee (a' \wedge b') = I$ (1)

Also

$(a \vee b) \wedge (a' \wedge b') = [(a \vee b) \wedge a'] \wedge b'$ (associative law)

$= [(a \wedge a') \vee (b \wedge a')] \wedge b'$ (distributive law)

$= [0 \vee (b \wedge a')] \wedge b'$ (complement law)

$= (b \wedge a') \wedge b'$ (identity law)

$= (b \wedge b') \wedge a'$ (associative law)

$= 0 \wedge a'$ (complement law)

$= 0$ (Boundedness law)

So we have

$(a \vee b) \wedge (a' \wedge b') = 0$ (2)

Hence from (1) and (2), we obtain $a' \wedge b'$ is complement of $a \vee b$, i.e. $(a \vee b)' = a' \wedge b'$.

4.14 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.
2 Seymour Lepschutz, Finite Mathematics (International edition 1983),

McGraw-Hili Book Company, New York.
3 C.L. Liu, Elements of Discrete Mathematics,
McGraw- Hili Book Co.
4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 5 **Written by Dr. Vizender Singh**

BOOLEAN ALGEBRA - II

Structure:

5.0 Learning Objectives

In this chapter the reader will learn some more results in Boolean algebra.

5.1 Introduction

In the previous chapter we have studied about some basic concepts of a Boolean algebra**.** In the present chapter we will study some more results in Boolean algebra namely, Boolean expressions, Sum- of- Products expressions, complete Sum- of- Products canonical forms, Boolean functions and their equivalence, logic gates and circuits (AND, OR and NOT gates), minimization of Boolean functions. Examples are also given to illustrate these topics.

5.2 Boolean Expressions

Definition: Let $x_1, x_2, \ldots\ldots, x_n$ be a set of n variables (or letters or symbols). **A Boolean Polynomial (Boolean expression, Boolean form or Boolean formula)** $p(x_1, x_2, \ldots\ldots, x_n)$ in the variables $x_1, x_2, \ldots\ldots, x_n$ is defined recursively as follows:

1. The symbols 0 and I are Boolean polynomials.

2. $x_1, x_2, ..., x_n$ are all Boolean polynomials.

3. If $p(x_1, x_2, ..., x_n)$ and $q(x_1, x_2, ..., x_n)$ are two Boolean polynomials, then so are

$$p(x_1, x_2, ..., x_n) \vee q(x_1, x_2, ..., x_n)$$

and

$$p(x_1, x_2, ..., x_n) \wedge q(x_1, x_2, ..., x_n)$$

4. If $p(x_1, x_2, ..., x_n)$ is a Boolean polynomial, then so is

$(p(x_1, x_2, ..., x_n))'$

5. There are no Boolean polynomials in the variables $x_1, x_2, ..., x_n$ other than those obtained in accordance with rules 1 to 4.

Thus, Boolean expression is an expression built from the variables given using Boolean operations $\vee$, $\wedge$ and $'$.

For example, for variables x, y, z , the expressions

$p_1(x, y, z) = (x \vee y) \wedge z$

$p_2(x, y, z) = (x \vee y') \vee (y \wedge 1)$

$p_3(x, y, z) = (x \vee (y' \wedge z)) \vee (x \wedge (y \wedge l))$

are Boolean expressions.

Notice that a Boolean expression in n variables may or may not contain all the n variables. Obviously, an infinite number of Boolean expressions may be constructed in n variables.

5.3 Literal, Fundamental Product and Minterm in a Boolean Expression

5.3.1 Definition: A **literal** is a variable or complemented variable such as x, x' , y, y' , and so on.

5.3.2 Definition: A **fundamental product** is a literal or a product of two or more literals in which no two literals involve the same variable.

For example,

$x \wedge z'$, $x \wedge y' \wedge z$, x , y' , $x' \wedge y \wedge z$

are fundamental products whereas

$x \wedge y \wedge x' \wedge z$ and $x \wedge y \wedge z \wedge y$

are not fundamental products.

5.3.3 Definition: A fundamental product which involves all the variables is called a **minterm or complete product**.

Remark: In what follows we shall denote $x \wedge y$ by $x\ y$.

Any product of literals can be reduced to either 0 or a fundamental product.

For example, consider $x\ y\ x'\ z$. Since $x \wedge x' = 0$ by complement law, so we have
$x\ y\ x'\ z = 0$.

Similarly, if we consider $x\ y\ z\ y$, then since $y \wedge y = y$ (using idempotent law), we have
$x\ y\ z\ y = x\ y\ z$, which is a fundamental product.

5.3.4 Definition: A fundamental product P_1 is said to be contained in (or included in) another fundamental product P_2 if the literals of P_1 are also literals of P_2.

For example, $x'\ z$ is contained in $x'\ y\ z$ but $x'\ z$ is not contained in $x\ y'\ z$ since x' is not a literal of $x\ y'\ z$.

We observe that if P_1 is contained in P_2, say $P_2 = P_1 \wedge Q$, then

$$P_1 \vee P_2 = P_1 \vee (P_1 \wedge Q)$$

$$= P_1 \qquad \text{(by absorption law)}$$

$$\therefore \quad P_1 \vee P_2 = P_1$$

For example,

$$x'\ z \vee x'\ y\ z = x'\ z$$

5.4 Sum-of-Products Boolean Expression and Algorithm for Finding Sum-of-Products Forms

5.4.1 Sum-of-Products Boolean Expression

Definition: A Boolean expression E is called a **sum-of-products expression (disjunctive Normal Form or DNF)** if E is a fundamental product or the sum (join) of two or more fundamental products none of which is contained in another.

Definition: Two Boolean expressions $P(x_1, x_2,.....,x_n)$ and $Q(x_1, x_2,.....,x_n)$ are called **equivalent (or equal)** if one can be obtained from the other by a finite number of applications of the identities of a Boolean algebra.

Definition: Let E be any Boolean expression. A sum of product form of E is an equivalent Boolean sum- of- products expression.

5.4.2 Algorithm for Finding Sum-of-Products Forms

The input is a Boolean expression E. The output is a sum-of-products expression equivalent to E.

Step 1. Use De Morgan's Law and involution law to move the complement operation into any parenthesis until finally the complement operation only applies to variables. Then E will consists only sums and products of literals.

Step 2. Use the distributive operation to next transform E into a sum of products.

Step 3. Use the commutative, idempotent and complement laws to transform each product in E into 0 or a fundamental product.

Step 4. Use the absorption law and identity law to finally transform E into a sum- of- products expression.

Example: Apply the above algorithm to the Boolean expression:

$$E = ((x\ y)'\ z)'\ [(x' + z)\ (y' + z')]'$$

Step 1. Using De Morgan's laws and involution law, we obtain

$$E = ((x\ y)'' \vee z')\ [(x' \vee z)' \vee (y' \vee z')']$$

$$= (x\ y \vee z') \wedge [(x \wedge z') \vee (y \wedge z)]$$

Thus E consists only of sum and products of literals.

Step 2. Using the distributive laws, we obtain

$$E = (x\ y + z')\ x\ z' + (x\ y + z')\ yz$$

$$= x\ y\ x\ z' + x\ z'\ z' + x\ y\ y\ z + y\ z\ z'$$

Thus E is now a sum of products.

Step 3. Using commutative, idempotent and complement law, we obtain

$$E = x\ y\ z' + x\ z' + x\ y\ z + 0$$

Thus each term in E is a fundamental product or 0.

Step 4. Using absorption law, we obtain

$$x\ z' + x\ y\ z' = x\ z' \vee (x\ z' \wedge y)$$

$$= x\ z'$$

Hence

$$E = x\ z' + x\ y\ z + 0$$

Step 5. Now using identity law, we obtain

$$E = x\ z' + x\ y\ z\ ,$$

which is the required sum-of-products expression.

5.5 Complete Sum-of-Product Boolean Expression and Algorithm for Obtaining Complete Sum- of- Product Expression

5.5.1 Complete Sum-of-Product Boolean Expression

Definition: A Boolean expression E (x_1, x_2, ….., x_n) is said to be a **complete sum-of-product expression (or full disjunctive normal form or disjunctive canonical form or the minterm**

canonical form) if E is a sum-of-products expression where each product involves all the n variables.

A fundamental product which involves all the variables is called a minterm and there is a maximum of 2^n such products for n variables.

It can be seen that **"every non- zero Boolean expression $E(x_1, x_2, \ldots\ldots, x_n)$ is equivalent to a complete sum-of-product expression and such a representation is unique."**

5.5.2 Algorithm for Obtaining Complete Sum- of- Product Expression

The input is a Boolean sum-of-products expression $E(x_1, x_2, \ldots.., x_n)$. The output is a complete sum-of-products expression equivalent to E.

Step 1. Find a product P in E which does not involve the variable x_i and then multiply P by $(x_i + x_i')$. Also deleting any repeated products (This is possible since $x + x' = I$ and $P + P = P$).

Step 2. Repeat step 1 until every product in E is a minterm, i.e. every product P involves all the variables.

5.6 Minimal Sum- of- Products Boolean Expression

Consider a Boolean sum-of-products expression E. Let E_L denote the number of literals in E (counted according to multiplicity) and let E_S denote the number of summands in E. For example, let

$$E = x\,y\,z' + x'\,y'\,z + x\,y'\,z'\,t + x'\,y\,z\,t$$

Then

$$E_L = 3 + 3 + 4 + 4 = 14 \qquad \text{and} \qquad E_S = 4.$$

Let E and F be equivalent Boolean sum-of-products expressions. Then E is called **simpler** than F if

(i) $E_L < F_L$ and $E_S \leq F_L$

or

(ii) $E_L \leq F_L$ and $E_S < F_L$

Definition: A Boolean sum-of-product expression is called minimal if there is no equivalent sum-of-product expression which is simpler than E.

There can be more than one equivalent minimal sum-of-products expressions.

5.7 Prime Implicants of a Boolean Expression

Definition: A fundamental product P is called **prime implicants** of a Boolean expression E if P + E = E but no other fundamental product contained in P has this property.

Example: Consider the following Boolean expression

$$E = x\,y' + x\,y\,z' + x'\,y\,z'$$

Then prove that $x\,z'$ is a prime implicant of E whereas x and z' are not prime implicants of E.

Solution: For this, we will show that $x\,z' + E = E$, $x + E \neq E$ and $z' + E \neq E$.

Now the complete sum-of-products form of Boolean expression E is

$$E = x\,y'(z + z') + x\,y\,z' + x'\,y\,z'$$

$$= x\,y'\,z + x\,y'\,z' + x\,y\,z' + x'\,y\,z' \qquad (1)$$

Then we find the complete sum-of-products form of $x\,z'$. So, we have

$$x\,z' = x\,z'(y + y')$$

$$= x\,z'\,y + x\,z'\,y' \qquad (2)$$

Also we know that the complete sum- of- products form is unique, $A + E = E$, where $A \neq 0$ if and only if the summands in the complete sum-of-products form for A are among the summands in the complete sum-of-products form for E. We observe that summands $x\,y\,z'$ and $x\,y'\,z'$ in (2) are in the complete form of E as given in (1).

Therefore, we have

$$x\,z' + E = E$$

Also, the complete sum-of-products form of x is

$$x = x\,(y + y')\,(z + z')$$

$$= (x\,y + x\,y')\,(z + z')$$

$$= x\,y\,z + x\,y\,z' + x\,y'\,z + x\,y'\,z'$$

The summand x y z of x is not a summand of E. Hence

$$x + E \neq E$$

Similarly, the complete sum-of-product form of z' is

$$z' = z'\,(x + x')\,(y + y')$$

$$= (z'\,x + z'\,x')\,(y + y')$$

$$= z'\,x\,y + z'\,x\,y' + z'\,x'\,y + z'\,x'\,y'$$

The summand $x'\,y'\,z'$ of z' is not a summand of E. Hence

$$z' + E \neq E$$

Thus the fundamental products x and z' contained in $x\,z'$ do not have the property $P + E = E$ where as $x\,z'$ has this property. Hence $x\,z'$ is a prime implicant of E whereas x and z' are not prime implicants of E.

It can be seen "a minimal sum-of-products form for a Boolean expression E is a sum of prime implicants of E".

5.8 Logic Gates and Circuits

5.8.1 Logic Circuits (Or Logic Networks)

Definition: Logic circuits (or logic networks) are structures which are built up from certain elementary circuits called logical gates.

5.8.2 Basic Logic Gates

There are three basic logic gates. The lines (wires) entering the gate symbol from the left are input lines and the single line on the right is the output line.

1. OR Gate: An OR gate has input x and y and output $z = x \vee y$ or $z = x + y$, where addition (or Join) is defined by the truth table. In this case the output $z = 0$ only when inputs $x = 0$ and $y = 0$.

The symbol and the truth table for OR gate are shown in the diagram below:

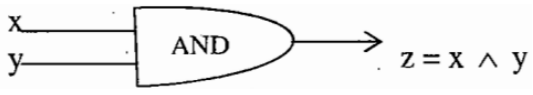
Truth Table for OR gate:

x	y	x + y
1	1	1
1	0	1
0	1	1
0	0	0

Thus output is 0 only when $x = 0$, $y = 0$, otherwise it is 1.

The OR gate may have more than two inputs. The output in such a case will be 0 if all the inputs are 0.

2. AND Gate: In this gate the inputs are x and y and output is $x \wedge y$ or $x \cdot y$ or xy, where multiplication is defined by the truth table.



Truth Table for AND gate:

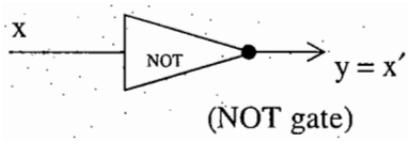
x	y	$z = x \wedge y$

1	1	1
l	0	0
0	1	0
0	0	0

Thus output is 1 only when x = 1, y = 1, otherwise it is zero.

The AND gate may have more than two inputs. The output in such a case will be 1 if all the inputs are 1.

3. NOT Gate (inverter): The diagram below shows NOT gate with input x and output $y = x'$, where inversion, denoted by the prime, is defined by the truth table:



(NOT gate)

Truth Table for NOT gate:

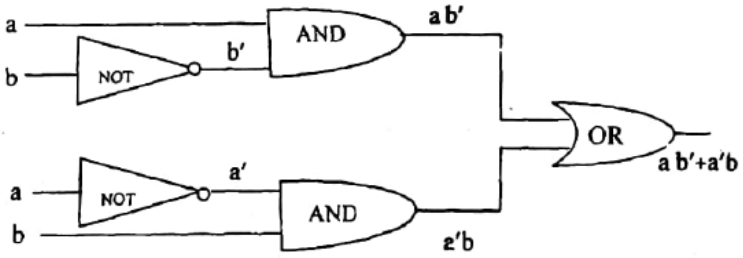
x	$y = x'$
1	0
0	1

For example, if x = 10101, then output x' in NOT gate shall be

$x' = 01010$

Example: Draw logic circuit for $a\,b' + a'\,b$

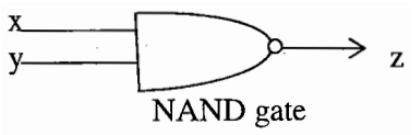
Solution: The logic circuit for the given expression is shown below



5.8.3 NAND and NOR Gates

NAND and NOR gates are frequently used in computers.

NAND gate: It is equivalent to AND gate followed by a NOT gate. Its symbol is



NAND gate

Its truth table is

x	y	x y	z = (x y)′
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Thus, the output of a NAND gate is 0 if and only if all the inputs are 1.

NOR gate: This gate is equivalent to OR gate followed by a NOT gate. Its symbol is

NOR Gate

Its truth table is as shown as:

x	y	x + y	(x + y)′

1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

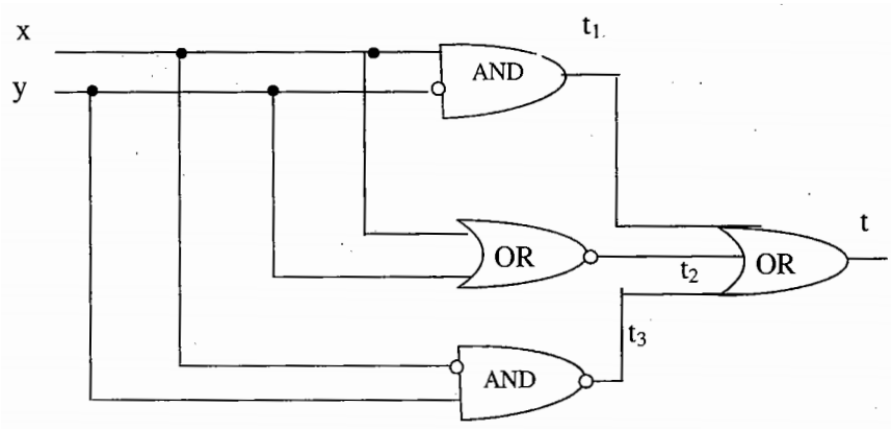
Thus, the output of NOR gate is 1 if and only if all inputs are 0.

5.8.4 Logic Circuits as a Boolean Algebra

The truth tables for OR, AND and NOT gates are respectively identical to the truth tables for the propositions $p \vee q$ (disjunction, "p or q"), $p \wedge q$ (Conjunction, "p and q") and $\sim p$ (negation, "not p"). The only difference is that 0 and 1 are used instead of F (contradiction) and T (tautology). Thus the logic circuits satisfy the same laws as do propositions and hence they form a Boolean algebra. Hence, we have established the following:

Theorem: Logic circuits form a Boolean algebra.

Example: Express the output of the logic circuit below as a Boolean expression. (Here small circle represents complement (NOT))



Solution: We see that

$$t_1 = x\, y'$$

$$t_2 = (x + y)'$$

$t_3 = (x' y)'$

So we have

$t = t_1 + t_2 + t_3$

$= x y' + (x + y)' + (x' y)'$

5.9 Boolean Function

We know that ordinary polynomials could produce functions by substitution. For example, the polynomial $x y + y z^3$ produces a function $f : \mathbf{R^3} \rightarrow \mathbf{R}$ by letting

$f(x, y, z) = x y + yz^3$.

Thus $f (3, 4, 2) = 3. 4 + 4. 2^3 = 44$. In a similar way, Boolean polynomials involving n variables produce functions from B_n to B.

Definition: Let $(B, +, ., ', 0, I)$ be a Boolean algebra. A function $f : B_n \rightarrow B$ which is associated with a Boolean expression (polynomial) is n variables is called a **Boolean function.**

Thus a Boolean function is completely determined by the Boolean expression $\alpha (x_1, x_2,...., x_n)$ because it is nothing but the evaluation function of the expression. It may be mentioned here that every function $g : B_n \rightarrow B$ need not be a Boolean function.

If we assume that the Boolean algebra B is of order 2^m for $m \geq 1$, then the number of functions from B_n to B is greater than 2^{2n} showing that there are functions from B_n to B which are not Boolean functions. On the other hand, for m = 1, that is, for a two element Boolean algebra, the number of functions from B_n to B is 2^{2n} which is same as the number of distinct Boolean expressions in n variables. Hence every function from B_n to B in this case is a Boolean function.

Example: Show that the following Boolean expressions are equivalent to one another. Obtain their sum-of-product canonical form.

(a) $(x + y) (x' + z) (y + z)$

(b) $(x z) + (x' y) + (y z)$

(c) $(x + y)(x' + z)$

(d) $x z + x' y$

Solution: The binary evaluation of the expressions are

x	y	z	x+y	x′+z	y+z	(a)	(c)	xz	x′y	yz	(b)	(d)
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	0	1	0	1	1
0	1	1	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	0	0	1	1
1	1	0	1	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	1	1

Since the values of the given Boolean expression are equal over every triple of the two element Boolean algebra, so they are equal.

To find the sum-of-product **canonical (complete)** form, we see that (d) is in sum-of-product form. Therefore to find complete sum-of-product form, we have

$$(d) = (x z) + (x' y)$$

$$= x z (y + y') + (x' y)(z + z')$$

$$= x z y + x z y' + x' y z + x' y z'$$

5.10 Method to Find Truth Table of a Boolean Function

Consider a logic circuit consisting of 3 input devices x, y, z. Each assignment of a set of three bits to the inputs x, y, z yields an output bit for t. There are $2^n = 2^3 = 8$ possible ways to assign bits to the input as follows:

000, 001, 010, 011, 100, 101, 110, 111.

The assumption is that the sequence of first bits is assigned to x, the sequence of second bits to y, and the sequence of third bits to z. Thus the above set of inputs may be rewritten in the form

x = 00001111, y = 00110011, z = 01010101

These three sequences (of 8 bits) contain the eight possible combinations of the input bits.

The truth table T = T(L) of the circuit L consists of the output t that corresponds to the input sequences x, y, z.

The truth table is same as we generally have written in vertical columns. The difference is that here we write x, y, z and t horizontally.

Consider a logic circuit L with n input devices. There are many ways to form n input sequences x_1, x_2, ..., x_n so that they contain 2^n different possible combinations of the input bits (Each sequence must contain 2^n bits).

The assignment scheme is:

x_1 : Assign 2^{n-1} bits which are 0 followed by 2^{n-1} bits which are 1.

x_2: Assign 2^{n-2} bits which are 0 followed by 2^{n-2} bits which are 1.

x_3 : Assign 2^{n-3} bits which are 0 followed by 2^{n-3} bits which are 1.

and so on.

The sequence obtained in this way is called **"Special Sequence"**. Replacing 0 by 1 and 1 by 0 in the special sequences yield the **complements** of the special sequences.

Example: Suppose a logic circuit L has n = 4 input devices x, y, z, t.
Then $2^n = 2^4 = 16$ bit special sequences for x, y, z, t are
x = 0000000011111111 ($2^{n-1} = 2^3$ = 8 zeros followed by 8 ones)

y = 0000111100001111 ($2^{n-2} = 2^{4-2}$ = 4 zeros followed by 4 ones)

z = 0011001100110011 ($2^{n-3} = 2^{4-3}$ = 2 zeros followed by 2 ones)

t = 0101010101010101 ($2^{n-4} = 2^{4-4} = 2^0$ = 1 zeros followed by 1 one)

5.10.1 Algorithm for Finding Truth Table for a Logic Circuit L where Output T is given by a Boolean Sum- of- Product Expression in the Input

The input is a Boolean sum-of-products expression $t(x_1, x_2,.....)$.

Step 1. Write down the special sequences for the inputs $x_1, x_2,....$ and their complements.

Step 2. Find each product appearing in $t(x_1, x_2, ...)$ keeping in mind that $x_1. x_2 .... = 1$ in a position if and only if all $x_1 , x_2.....$ have 1 in the position.

Step 3. Find the sum t of the products keeping in mind that $x_1 + x_2 +..... = 0$ in a position if and only if all $x_1, x_2,.....$ have 0 in the position.

5.11 Minimization of Boolean Expressions

There are two ways to minimize Boolean Expressions:

(i) Algebraic Method (ii) Karnaugh Map

5.11.1 Algebraic Method

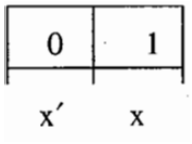
5.11.2 Karnaugh Map

Karnaugh Map is a graphical procedure to represent Boolean function as an "or" combination of minterms where minterms are represented by squares. This procedure is easy to use with functions $f : B_n \rightarrow B$, if n is not greater than 6. We shall discuss this procedure for n = 2, 3 and 4.

A Karnaugh map structure is an area which is subdivided into 2^n cells, one for each possible input combination for a Boolean function of n variables. Half of the cells are associated with an input value of 1 for one of the variables and the other half are associated with an input value of 0 for the same variable. This association of cell is done for each variable, with the splitting of the 2^n cells yielding a different pair of halves for each distinct variable.

Case of 1 variable:

In this case, the Karnaugh map consists of $2^1 = 2$ squares.

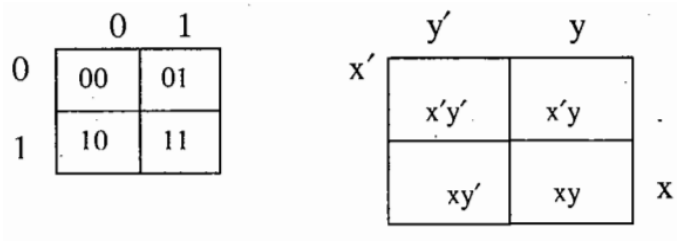


The variable x is represented by the right square and its complement x′ by the left square.

Case of 2 variables:

For n = 2, the Boolean function is of two variables, say x and y. We have $2^2 = 4$ squares, that is, a 2 × 2 matrix of squares. Each square contains one possible input from B_2.

The variable x appears in the first row of the matrix as x′ whereas x appears in the second row as x. Similarly, y appears in the first column as y′ and as y in the second column.



(two variables Karnaugh Map)

In this case, x is represented by the points in lower half of the map and y is represented by the points in the right half of the map.

Definition: Two fundamental products are said to be **adjacent** if they have the same variables and if they differ in exactly one literal. Thus there must be an uncomplemented variable in one product which is complemented in the other.

For example, if $P_1 = x\ y\ z'$ and $P_2 = x\ y'\ z'$, then they are adjacent.

The sum of two such adjacent products will be a fundamental product with one less literal.

For example, in the case of above mentioned adjacent products,

$$P_1 + P_2 = x\ y\ z' + x\ y'\ z'$$

$$= x\ z'\ (y + y')$$

$= x\, z' (1)$

$= x\, z'$

We note that two squares in Karnaugh map above are adjacent if and only if squares are geometrically adjacent, that is, have a side in common.

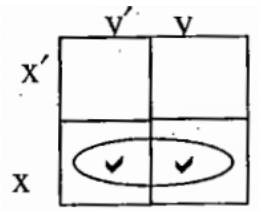
We know that a complete sum-of-products Boolean expression E(x, y) is a sum of minterms and hence can be represented in the Karnaugh map by placing checks in the appropriate square. A prime implicant of E(x, y) will be either a **pair of adjacent squares** in E or an isolated square (a square which is not adjacent to other squares of E(x, y)). A minimal sum- of- products form for E(x, y) will consists of a minimal number of prime implicants which cover all the squares of E(x, y).

Example: Find the prime implicants and a minimal sum-of-products form for each of the following complete sum-of-products Boolean expression:

(a) $E_1 = x\, y + x\, y'$ (b) $E_2 = x\, y + x'\, y + x'\, y'$

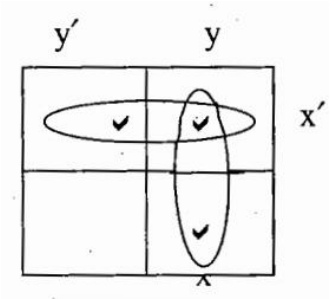
(c) $E_3 = x\, y + x'\, y'$

Solution: (a) The Karnaugh map for E_1 is



Check the squares corresponding to x y and x y′ . Here we see that E_1 consists of one prime implicant, which is the two adjacent squares designated by the loop. The pair of adjacent squares represents the variable x. So, x is the only prime implicant of E_1. Consequently, $E_1 = x$ is its minimal sum.

(b) The Karnaugh map for E_2 is

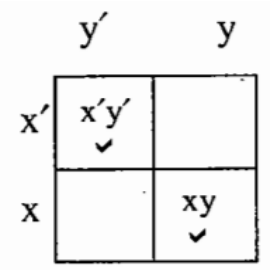


Check the squares corresponding to x y, x′ y, x′ y′ . The expression E_2 contains two pairs of adjacent squares (designated by two loops) which include all the squares of E_2. The vertical pair represents y and the horizontal pair represents x′ . Hence y and x′ are the prime implicants of E_2.Thus

$$E_2(x, y) = x' + y$$

is minimal sum.

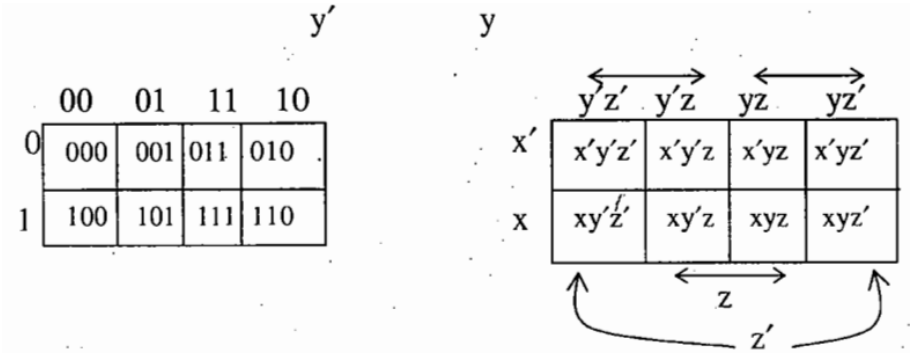
(c) The Karnaugh map for E_3 is



Check (tick) the squares corresponding to x y and x′ y′ . The expression E_3 consists of two isolated squares which represent x y and x′ y′ . Hence x y and x′ y′ are the prime implicants of E_3 and so E_3 = x y + x′ y′ is its minimal sum.

Case of 3 variables:

We now consider the case of a function f : $B_3 \rightarrow B$ which is function of x, y and z. The Karnaugh map corresponding to Boolean expression E(x, y, z) is shown in the diagram below:



Here x, y, z are respectively represented by lower half, right half and middle two quarters of the map.

Similarly, x′ , y′ , z′ are respectively represented by upper half, left half, and left and right quarter of the map.

Definition: By a **Basic Rectangle** in the Karnaugh map with three variables, we mean a square, two adjacent squares and four squares which form a one- by- four, or a two- by- two rectangle. These basic rectangles correspond to fundamental products of three, two and one literal respectively.

Further, the fundamental product represented by a basic rectangle is the product of just those literals that appear in every square of the rectangle.

Let a complete sum- of- products Boolean expression E(x, y, z) is represented in the Karnaugh map by placing checks in the appropriate squares. A prime implicant of E will be a maximal basic rectangle of E, i.e., a basic rectangle contained in E which is not contained in any larger basic rectangle in E.

A minimal sum-of-products form for E will consist of a minimal cover of E, i.e., a minimal number of maximal basic rectangles of E which together include all the squares of E.

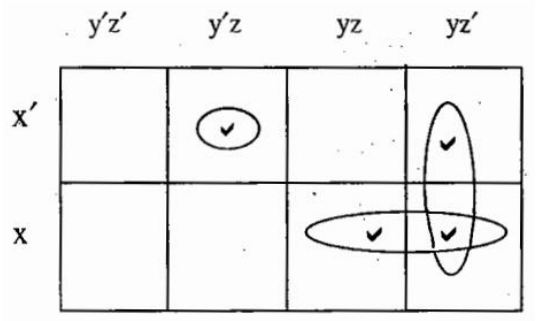
Example: Find the prime implicants and a minimal sum-of-products form for each of the following complete sum- of- products Boolean expressions:

(a) $E_1 = x\,y\,z + x\,y\,z' + x'\,y\,z' + x'\,y'\,z$

(b) $E_2 = x\,y\,z + x\,y\,z' + x\,y'\,z + x'\,y\,z + x'\,y'\,z$

(c) $E_3 = x\,y\,z + x\,y\,z' + x'\,y\,z' + x'\,y'\,z + x'\,y'\,z'$

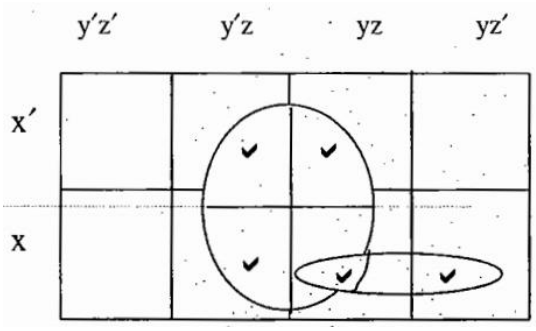
Solution: (a) The Karnaugh map for E_1 is



We check the four squares corresponding to four summands in E_1. Here E_1 has three prime implicants (maximal basic rectangles) which are encircled. These are x y, y z′ and x′ y′ z. All three are needed to cover E_1. Hence minimal sum for E_1 is

$E_1 = x\,y + y\,z' + x'\,y'\,z$

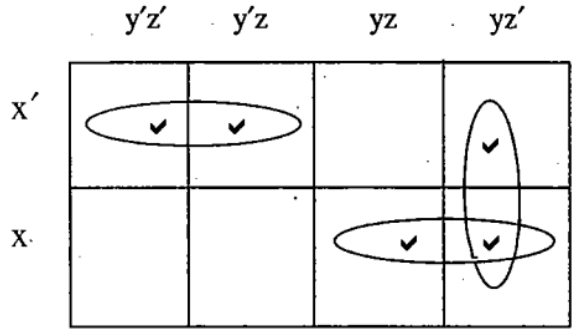
(b) The Karnaugh map for E_2 is



Check the squares corresponding to the five summands. E_2 has two prime implicants which are circled. One is the two adjacent squares which represent x y, and the other is the two-by-two square which represents z. Both are needed to cover E_2, so the minimal sum for E_2 is

$E_2 = x\,y + z$

(c) The Karnaugh map for E_3 is



Check the squares corresponding to the five summands. Here E_3 has three prime implicants x y, y z′ , x′ y′ . All these are needed in a minimal cover of E_3. Hence E_3 has minimal sum as

$$E_3 = x\,y + y\,z' + x'\,y'$$

Remark: To find the fundamental product represented by a basic rectangle, find literals which appear in all the squares of the rectangle.

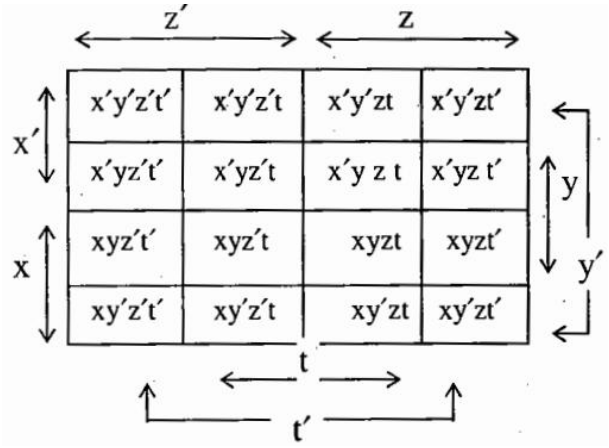
Case of 4 Variables:

We consider a Boolean function $f : B_4 \rightarrow B$, considered as a function of x, y, z and t. Each of the 16 squares (2^4) corresponds to one of the minterms with four variables.

$$x\,y\,z\,t,\ x\,y\,z\,t',\ldots\ldots\ldots\ldots,\ x'\,y\,z'\,t$$

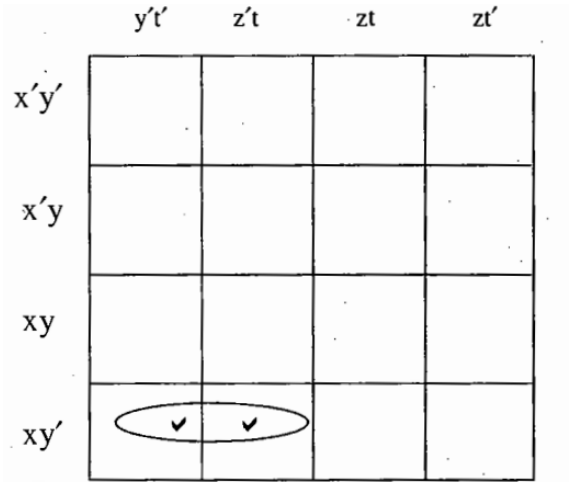
We consider first and last columns to be adjacent, and first and last rows to be adjacent, both by Wrap around, and we look for rectangles with sides of length some power of 2, so the length is 1, 2 or 4. The expression for such rectangles is given by intersecting the large labelled rectangles.

	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010



A basic rectangle in a four variable Karnaugh map is a square, two adjacent squares, four squares which form a one-by-four or two- by- two rectangles, eight square squares which form a two- by- four rectangle. These rectangles correspond to fundamental product with four, three, two and one literal respectively. Maximal basic rectangles are prime implicants.

Example: Find the fundamental product P represented by the basic rectangle in the Karnaugh map given below:



Solution: We find the literals which appear in all the squares of the basic rectangle. Then P will be the product of such literals.

Here x, y′ , z′ appear in both squares. Hence

$$P = x\, y'\, z'$$

is the fundamental product represented by the basic rectangle in the given Karnaugh map.

5.12 Check Your Progress

1. Express $x_1 \vee x_2$ in its complete sum-of-products form in three variables x_1, x_2, x_3.

Solution: We have, using the above stated algorithm,

$x_1 + x_2 = [x_1 (x_2 + x_2')] + [x_2 (x_1 + x_1')]$ (complement law)

$= x_1 x_2 + x_1 x_2' + x_2 x_1 + x_1' x_2$ (distributive law)

$= x_1 x_2 + x_1 x_2' + x_1' x_2$ (idempotent law)

= ----------

(complement law)

$= x_1 x_2 x_3 + x_1 x_2 x_3' + x_1 x_2' x_3 + x_1 x_2' x_3' + x_1' x_2 x_3 + x_1' x_2 x_3'$ (distributive law)

which is the complete sum-of-products form in x_1, x_2, x_3.

2. Consider the expression

$E_1(x, y, z) = x z' + y' z + x y z'$

Although the expression E_1 is a sum of products, it is not a sum-of-products expression because the product $x z'$ is contained in the product $x y z'$. But, by absorption law, E_1 can be expressed as

$E_1 (x, y, z) = x z' + y' z + x y z'$

=---------- (by commutative law)

$= x z' + y' z$ (by absorption law)

which is a sum-of-product form for E_1.

3. Express each Boolean expression E(x, y, z) as a sum-of-products and then in its complete sum-of-products Form E = z(x+ y) + y.

4. Let $E = xy' + xyz' + x'yz'$ Prove that $xz' + E = E$.

5. Let $E = xy + y't + x'yz' + xy'zt'$. Find minimal sum for E.

5.13 Summary

In this chapter we have studied some more results in Boolean algebra namely, Boolean expressions, Sum- of- Products canonical forms, Boolean functions and their Equivalence, logic gates and circuits (AND, OR and NOT gates), Minimization of Boolean Functions.

5.14 Keywords

Boolean expressions, Sum- of- Products forms, logic gates and circuits

5.15 Self-Assessment Test

1. Let a, b, c be any element in a Boolean algebra B.

(i) $a + a = a$ (ii) $a * a = a$

2. Let a be any element of a Boolean algebra B, If $a + x = 1$ and $a * x = 0$, then $x = a$.

3. Reduce the following Boolean products to either 0 or a fundamental product:
(a) xyx'z; (b) xyzy; (c) xyz'yx; (d) xyz'yx'z'.

4. Express each Boolean expression E(x, y, z) as a sum-of-products and then in its complete sum-of-products form E = x(xy + xy + yz).

5. Express E(x, y, z) = (x' + y)' + x'y in its complete sum-of-products form.

5.16 Answers to check your progress

1. $x_1 x_2 (x_3 + x_3') + x_1 x_2' (x_3 + x_3') + x_1' x_2 (x_3 + x_3')$

2. $x z' + x y z' + y' z$

3. First we have
E = z(x' + y) + y'= x'z + yz + y'. Then
E = x'z + yz + y' = x'z(y + y') + yz(x + x') + y'(x + x')(z + z')
= x'yz + x'y'z + xyz + x'yz + xy'z + xy'z'+ x'y'z + x'y'z
= xyz + xy'z + xy'z' + x'yz + x'y'z + x'y'z'.

4. Since the complete sum-of-products form is unique, $A + E = E$, where $A \_= 0$, if and only if the summands in the complete sum-of-products form for A are among the summands in the complete sum-of products form for E. Hence, first find the complete sum-of-products form for E:

$$E = xy'(z + z') + xyz' + x'yz'$$
$$= xy'z + xy'z' + xyz' + x'yz'$$

Express xz' in complete sum-of-products form xz' = xz' (y + y') = xyz' + xy' z'
Since the summands of xz are among those of E, we have xz' + E = E.

5. Write each prime implicant in complete sum-of-products form and then delete one by one those which are superfluous, i.e. those whose summands appear among the other summands. This finally yields E = y't + xz + yz' as a minimal sum for E.

5.17 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.

2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.

3 C.L. Liu, Elements of Discrete Mathematics, McGraw-Hili Book Co.

4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 6 **Written by Dr. Vizender Singh**

GRAPH THEORY – I

Structure:

6.0 Learning Objectives

The main objective of this chapter is to familiarize the students with the basic aspects of graph theory. The reader will learn about some basic definitions of graph theory and special graphs, namely regular graph, complete and complete bipartite graphs. The reader will also study walks, paths and circuits, connected and disconnected graphs, connected components of a graph. Examples are also given to illustrate these topics.

6.1 Introduction

In this chapter we will study graph which is pictorial representation of relations on sets.

Graphs are one of the prime objects of study in Discrete Mathematics. Graph theory is one of the branches of modern mathematics having experienced a most impressive development in recent years. In the beginning, Graph theory was only a collection of recreational or challenging problems like Euler tours or the four colouring of a map, with no clear connection among them, or among techniques used to attach them. The aim was to get a "yes" or "no" answer to simple existence questions. Under the impulse of Game Theory, Management Sciences and Transportation Network Theory, the main concern shifted to the maximum size of entities attached to a graph. It is no coincidence that graph theory has been independently discovered many times, since it may quite properly be regarded as an area of applied mathematics. The basic combinatorial nature of Graph theory and a clue to its wide applicability are indicated in the words of Sylvester, ''The theory of ramification is one of the pure colligation, for it takes no account of magnitude or position ; geometrical lines are used, but have no more real bearing on the matter than those employed in genealogical tables have in explaining the laws of procreation". Indeed, the earliest recorded mention of the subject occurs in the works of Euler, and although the original problem he was considering might be regarded as a somewhat frivolous puzzle, it did arise from the physical world. Subsequent rediscoveries of graph theory by Kirchhoff and Cayley also had their roots in the physical world.

Some Basic Definitions and Examples

Definition: A **graph** G (V, E) is a mathematical structure consisting of two finite sets V and E. The elements of V are called Vertices (or nodes) and the elements of E are called Edges. Each edge is associated with a set consisting of either one or two vertices called its endpoints.

The correspondence from edges to endpoints is called **edge-endpoint function**. This function is generally denoted by γ. Due to this function, some author denote graph by $G = (V, E, \gamma)$.

Definition: A graph consisting of one vertex and no edges is called a **trivial graph.**

Definition: A graph whose vertex and edge sets are empty is called a **null graph.**

Definition: An edge with just one end point is called a **loop** or a **self loop**. Thus, a loop is an edge that joins a single endpoint to itself.

Definition: An edge that is not a self-loop is called a **proper edge.**

Definition: If two or more edges of a graph G have the same vertices, then these edges are said to be **parallel** or **multi-edges**.

Definition: Two vertices that are connected by an edge are called **adjacent**.

Definition: An endpoint of a loop is said to be **adjacent to itself**.

Definition: An edge is said to be **incident** on each of its endpoints.

Definition: Two edges incident on the same endpoint are called **adjacent edges.**

Definition: The number of edges in a graph G which are incident on a vertex is called the **degree of that vertex**.

Definition: A vertex of degree zero is called an **isolated vertex**. Thus, a vertex on which no edges are incident is called isolated.

Definition: A graph without multiple edges (parallel edges) and loops is called **simple graph.**

Notation: In pictorial representations of a graph, the vertices will be denoted by dots and edges by line segments.

Example 1: Let

$$V = \{1, 2, 3, 4\} \text{ and } E = \{e_1, e_2, e_3, e_4, e_5\}.$$

Let γ be defined by

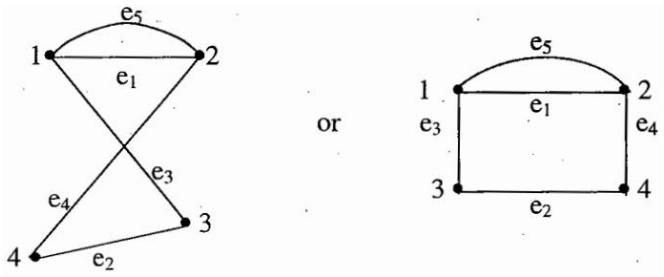
$\gamma(e_1) = \gamma(e_5) = \{1, 2\}$

$\gamma(e_2) = \{4, 3\}$

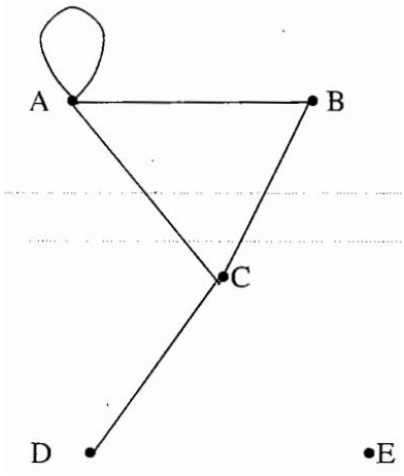
$\gamma(e_3) = \{1, 3\}$

$\gamma(e_4) = \{2, 4\}$

We note that both edges e_1 and e_5 have same endpoints $\{1, 2\}$. The endpoints of e_2 are $\{4, 3\}$, the endpoints of e_3 are $\{1, 3\}$ and endpoints of e_4 are $\{2, 4\}$. Thus the graph is



Example 2: Consider the graph with the vertices A, B, C, D and E pictured in the figure below.



In this graph, we see that

No. of edges = 5

Degree of vertex A = 4

Degree of vertex B = 2

Degree of vertex C = 3

Degree of vertex D = 1

Degree of vertex E = 0

Sum of the degree of vertices = 4 + 2 + 3 + 1 + 0 =10

Thus, we observe that

$$\sum_{i=1}^{5} \deg(v_i) = 2e ,$$

where $\deg(v_i)$ denotes the degree of vertex v_i and e denotes the number of edges.

6.2 Euler's Theorem (or The First Theorem of Graph Theory)

The sum of the degrees of the vertices of a graph G is equal to twice the number of edges in G. **Thus, total degree of a graph is even.**

Proof: Each edge in a graph contributes a count of 1 to the degree of two vertices (end points of the edge), that is, each edge contributes 2 to the degree sum. Therefore the sum of degrees of the vertices is equal to twice the number of edges.

Corollary: There must be an even number of vertices of odd degree in a given graph G.

Proof: We know, by the **Euler's Theorem**, that

$$\sum_{i=1}^{n} \deg(v_i) = 2 \times \text{no. of edges}$$

Thus the right hand side is an even number. Hence to make the left-hand side an even number there can be only even number of vertices of odd degree.

Remarks:

(i) A vertex of degree d is also called a **d- valent vertex.**

(ii) The degree (or valence) of a vertex 'v' in a graph G is the number of proper edges incident on 'v' plus twice the number of self- loops.

Theorem: A non-trivial simple graph G must have at least one pair of vertices whose degrees are equal.

Proof: Let the graph G has 'n' vertices. Then there appear to be 'n' possible degree values, namely 0, 1, ..., n – 1. But there cannot be both a vertex of degree 0 and a vertex of degree n – 1 because if there is a vertex of degree 0 then each of the remaining n– 1 vertices is adjacent to atmost n – 2 other vertices. Hence the n vertices of G can realize atmost n–1 possible values for their degrees. Hence the pigeonhole principle implies that at least two of the vertices have equal degree.

6.3 Simple Graph

Definition: A graph G is said to simple if it has no parallel edges or loops. In a simple graph, an edge with endpoints v and w is denoted by {v, w}.

Definition: For each integer n ≥ 1, let D_n denote the graph with n vertices and no edges. Then D_n is called the discrete graph on n vertices.

For example, we have

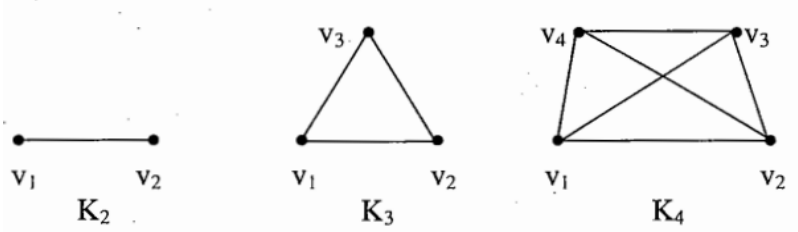
• • • and • • • • •

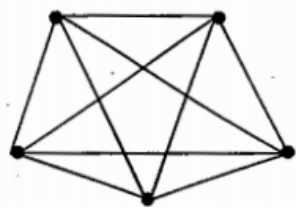
D_3 D_5

6.4 Complete Graph

Definition: Let n ≥ 1 be an integer. Then a simple graph with n vertices in which there is an edge between each pair of distinct vertices is called the **complete graph** on n vertices. It is denoted by K_n.

For example, the complete graphs K_2, K_3, K_4 and K_5 are shown in the figures below:





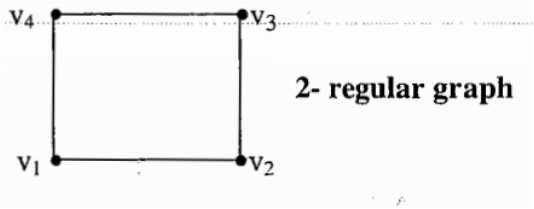
K_5

6.5 Regular Graph

Definition: If each vertex of a graph G has the same degree as every other vertex, then G is called a **regular graph**.

A **k-regular graph** is a regular graph whose common degree is k.

For example, consider K_3. The degree of each vertex in K_3 is 2. Hence K_3 is regular. Similarly, K_4 is regular. Also the graph shown below is regular because degree of each vertex here is 2.



But this graph is not complete because v_2 and v_4 have not been connected through an edge. Similarly, v_1 and v_3 are not connected by any edge.

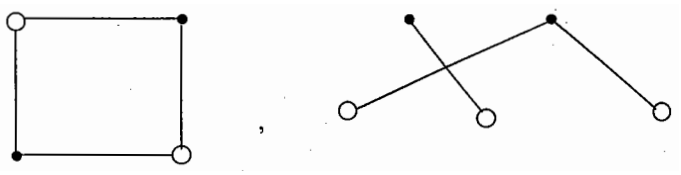
Thus, **a complete graph is always regular but a regular graph need not be complete.**

6.6 Bipartite Graph

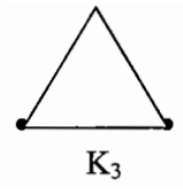
Definition: A **bipartite graph** G is a graph whose vertex set V can be partitioned into two subsets U and W, such that each edge of G has one endpoint in U and one endpoint in W.

The pair (U, W) is called a **Vertex bipartition of G** and the subsets U and W are called the bipartition subsets. Obviously, a bipartite graph cannot have any self loop.

Example 1: If Vertices in U are solid vertices and vertices in W are hollow vertices, then the following graphs are bipartite graphs:



Example 2: The smallest possible simple graph that is not bipartite is the complete graph K_3 shown below:



6.7 Complete Bipartite Graph

Definition: A complete bipartite graph G is a simple graph whose vertex set V can be partitioned into two subsets U = $\{v_1, v_2, \ldots, v_m\}$ and W =$\{w_1, w_2, .., w_n\}$such that for all i, k in {1, 2,....,m } and j, *l* in {1, 2,... n}

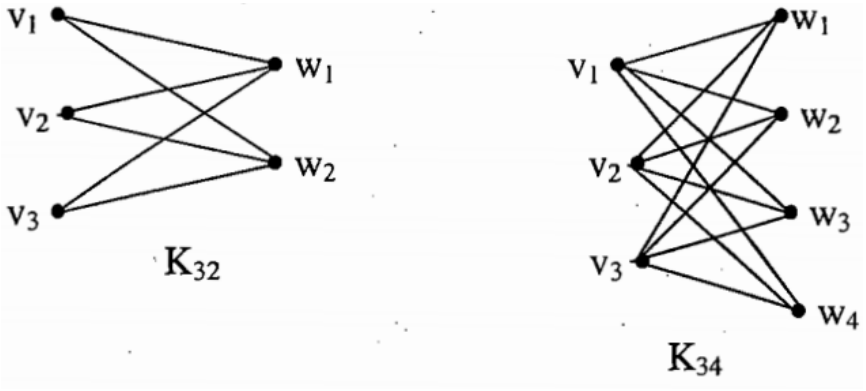
(i) there is an edge from each vertex v_i to each vertex w_j.

(ii) there is not an edge from any vertex v_i to any other vertex v_k.

(iii) there is not an edge from any vertex w_j to any other vertex w_l.

A complete bipartite graph on (m, n) vertices is denoted by $K_{m,n}$.

Example: The complete bipartite graphs $K_{3,2}$ and $K_{3,4}$ are shown in the figure below:



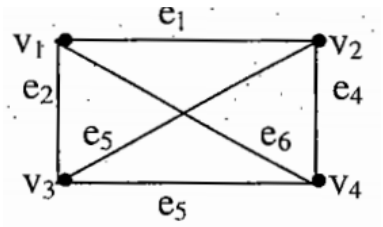
6.8 Subgraphs

6.8.1 Subgraph of a Graph

Definition: A graph H is said to be a subgraph of a graph G if and only if every ve1tex in H is also a vertex in G, every edge in H is also an edge in G and every edge in H has the same endpoints as in G.

We may also say that G is a supergraph of H.

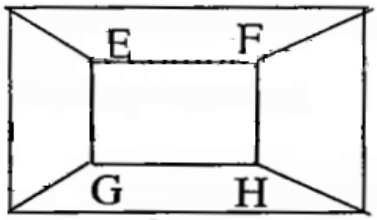
For example, consider the graph G_1



G_1

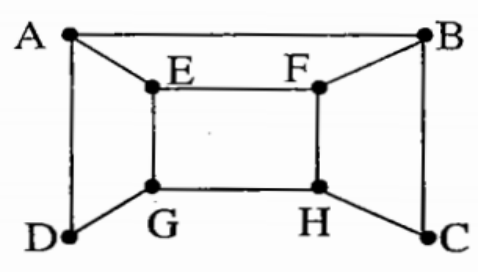
Then the graphs given below are the subgraphs of G_1

Similarly, consider the graph G_2



G_2

Then the graph given below is a subgraph of G_2:



6.8.2 Proper Subgraph

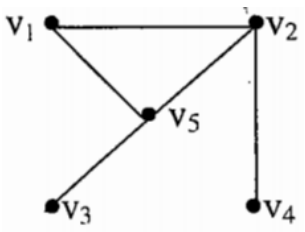
Definition: A subgraph H is said to be a **proper subgraph** of a graph G if vertex set V_H of H is a proper subset of the vertex set V_G of G or edge set E_H is a proper subset of the edge set E_G.

For example, the subgraphs in the above examples are proper subgraphs of the given graphs.

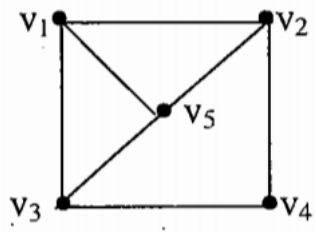
6.8.3 Spanning Subgraph

Definition: A subgraph H is said to span a graph G if $V_H = V_G$. Thus H is a spanning sub graph of graph G if it contains all the vertices of G.

For example the subgraph



spans the graph

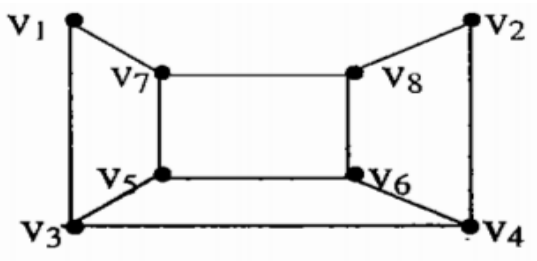


6.8.4 Complement of a Subgraph

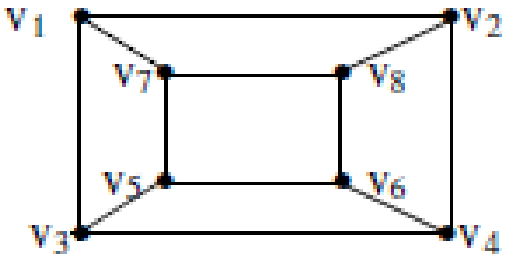
Definition: Let G = (V, E) be a graph. Then the **complement of a subgraph** G' = (V', E') with respect to the graph G is another subgraph G" = (V", E") such that E" = E – E' and V" contains only the vertices with which the edges in E" are incident.

For example, the subgraph

is the complement of the subgraph



with respect to the graph G shown in the figure below:



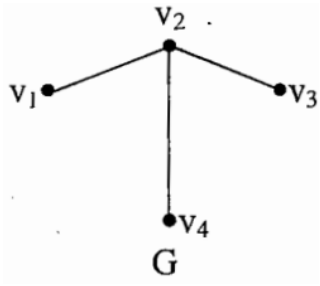
6.8.5 Complement of a Graph

Definition: If G is a simple graph, the **complement of G, (Edge complement),** denoted by G′ or G^c is a graph such that

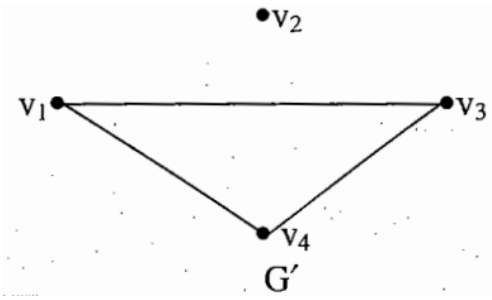
(i) The vertex set of G′ is identical to the vertex set of G, that is $V_{G'} = V_G$

(ii) Two distinct vertices v and w of G′ are connected by an edge if and only if v and w are not connected by an edge in G.

For example, consider the graph G

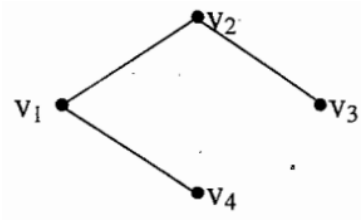


Then complement G′ of G is the graph

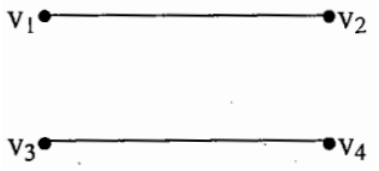


Example: Find the complement of the graphs:

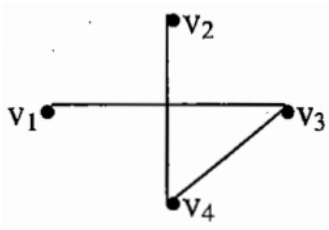
(a)



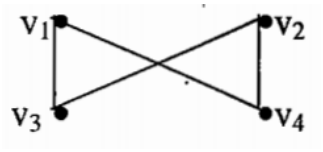
(b)



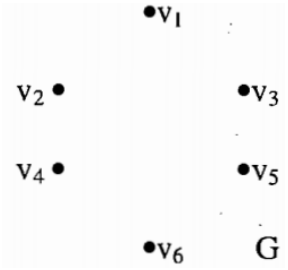
Solution: (a)



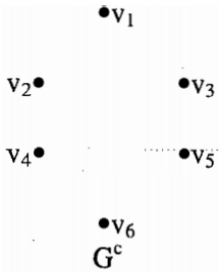
(b)



Example: Find the edge complement of the graph G shown below:



Solution: The edge complement of G is the following graph G^c



6.9 Walks, Paths and Circuits

Definition: In a graph G, a **walk** from vertex v_0 to vertex v_n is a finite alternating sequence:

$$\{v_0, e_1, v_1, e_2, \ldots\ldots v_{n-1}, e_n, v_n\}$$

of vertices and edges such that v_{i-1} and v_i are the endpoints of e_i.

The **trivial walk** from a vertex v to v consists of the single vertex v.

Definition: In a graph G, a **path** from the vertex v_0 to the vertex v_n is a walk from v_0 to v_n that does not contain a repeated edge.

Thus a **path** from v_0 to v_n is a walk of the form

$$\{v_0, e_1, v_1, e_2, v_2, \ldots\ldots v_{n-1}, e_n, v_n\}$$

where all the edges e_1 are distinct.

Definition: In a graph, a **simple path** from v_0 to v_n is a path that does not contain a repeated vertex.

Thus a simple path is a walk of the form

$$\{v_0, e_1, v_1, e_2, v_2, \ldots\ldots v_{i-1}, e_n, v_n\}$$

where all the e_i are distinct and all the v_i are distinct.

Definition: A walk in a graph G that starts and ends at the same vertex is called a **closed walk.**

Definition: A closed walk that does not contain a repeated edge is called a **circuit**.

Thus, a closed path is called a circuit (or a cycle) and so a circuit is a walk of the form

$$\{v_0, e_1, v_1, e_2, v_2, \ldots\ldots v_{n-1}, e_n, v_n\}$$

where $v_0 = v_n$ and all the e_i are distinct.

Definition: A **simple circuit** is a circuit that does not have any other repeated vertex except the first and the last.

Thus, a simple circuit is a walk of the form

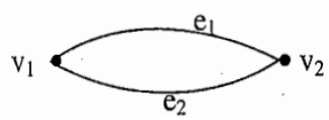
$\{v_0, e_1, v_1, e_2, \ldots\ldots v_{n-1}, e_n, v_n\}$

where all the e_1 are distinct and all the v_j are distinct except that $v_0 = v_n$.

Definition: In a graph the number of edges in the path $\{v_0, e_1, v_1, e_2, \ldots\ldots, e_n, v_n\}$ from v_0 to v_n is called the **length of the path.**

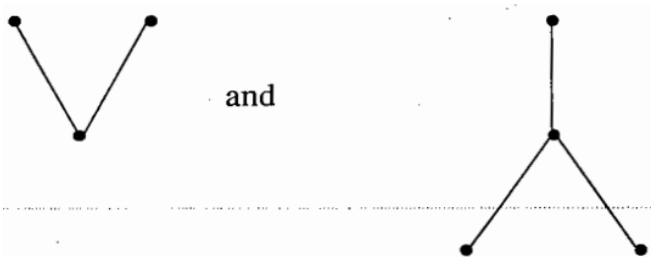
Definition: A cycle with k- edges is called a **k-cycle** or **cycle of length k**.

For example, loop is a cycle of length 1. On the other hand, a pair of parallel edges e_1 and e_2, shown below, is a cycle of length 2



Definition: A graph is said to be **acyclic** if it contains no cycle.

For example, the graphs



are acyclic.

Theorem: If there is a path from vertex v_1 to v_2 in a graph with n vertices, then there does not exist a path of more than n – 1 edges from vertex v_1 to v_2.

Proof: Suppose there is a path from v_1 to v_2. Let

$v_1, \ldots\ldots\ldots, v_i, \ldots\ldots\ldots, v_2$

be the sequence of vertices which the path meets between the vertices v_1 and v_2. Let there be m edges in the simple path. Then there will be m + 1 vertices in the sequence. Therefore if m > n–1, then there will be more than n vertices in the sequence. But the graph is with n vertices.

Therefore some vertex, say v_k, appears more than once in the sequence. So the sequence of vertices shall be

$v_1,\ldots\ldots\ldots,v_j,\ \ldots\ldots\ldots,v_k,\ldots\ldots\ldots,v_k,\ \ldots\ldots\ldots\ldots,v_2.$

Deleting the edges in the path that lead v_k back to v_k , we have a path from v_1 to v_2 that has less edges than the original one. This argument is repeated untill we get a path that has n– 1 or less edges.

Definition: Two vertices v_1 and v_2 of a graph G are said to be **connected** if and only if there is a walk from v_1 to v_2.

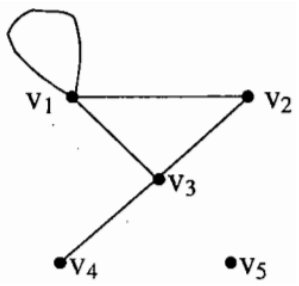
6.10 Connected and Disconnected Graphs

Definition: A graph G is said to be **connected** if and only if given any two vertices v_1 arid v_2 in G, there is a walk from v_1 to v_2. Thus, a graph G is connected if there exists a walk between every two vertices in the graph.

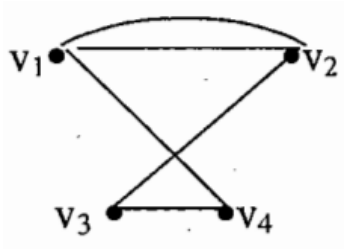
Definition: A graph which is not connected is called **Disconnected Graph**.

Example: Which of the graphs shown below are connected?

(a)



(b)



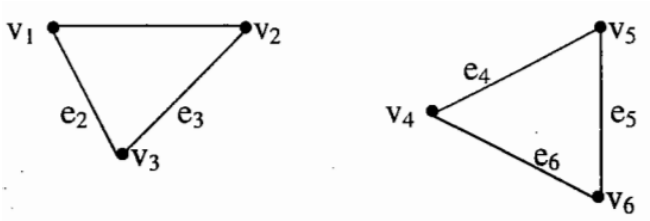
Solution: Graph (a) is not connected as there is no walk from any of v_1, v_2, v_3, v_4 to the vertex v_5.

The graph (b) is clearly connected.

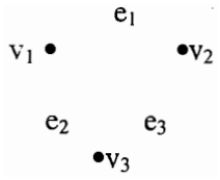
6.11 Connected Components of a Graph

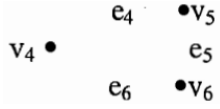
Definition: If a graph G is disconnected, then the various connected pieces of G are called the **connected components of the graph**.

Example: Consider the graph given below:

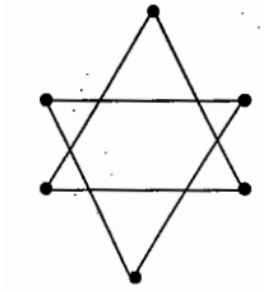


The graph is disconnected and has two connected components:

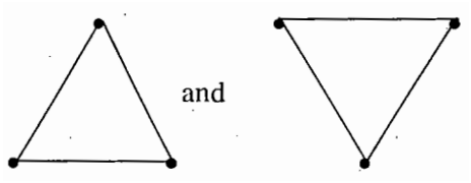
H_1 :  with vertex set $\{v_1, v_2, v_3\}$ and edge set $\{e_1, e_2, e_3\}$

H_2 :  with vertex set $\{v_4, v_5, v_6\}$ and edge set $\{e_4, e_5, e_6\}$

Example: Find the number of connected components in the graph



Solution: The connected components are:



Remark: If a connected component has n vertices, then degree of any vertex cannot exceed n – 1.

6.12 Check Your Progress

1. A finite connected graph G is Eulerian if and only if each vertex has even degree.

2. Let G be a finite graph with $n \geq 1$ 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.

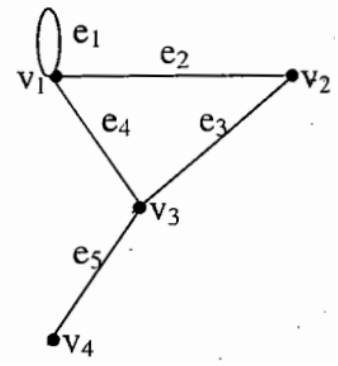
3. Prove $V - E + R = 2$.

4. Fill in the blanks
Let G be a finite connected planar graph with at least three vertices. Show that G has at least one vertex of degree 5 or less.
Sol. Let p be the number of vertices and q the number of edges of G, and suppose $\deg(u) \geq 6$ for each vertex u of G. But $2q$ equals the sum of the degrees of the vertices of G (Theorem 8.1); so $2q \geq 6p$. Therefore

---

This contradicts Theorem 8.9. Thus some vertex of G has degree 5 or less.

5. Consider the graph shown below



We note that

(i) The walk e_3, e_5 is a----------.

(ii) The walk e_1, e_2, e_3, e_5 is a path but it is not a simple path because the vertex v_1 is repeated (e_1 being a self- loop).

(iii) The walk e_2, e_3, e_4 is a ----------.

(iv) The walk e_2, e_3, e_4, e_1 is a circuit but it is not simple circuit because vertex v_1, repeats twice (or we may write that v_1 is met twice).

6.13 Summary

In this chapter we have studied some basic definitions of graph theory and special graphs, namely regular graph, complete and complete bipartite graphs. We have also discussed the topics walks, paths and circuits, connected and disconnected graphs, connected components of a graph along with suitable examples.

6.14 Keywords

Graph, Degree of a vertex, complete and complete bipartite graphs, walks, paths and circuits, connected and disconnected graphs

6.15 Self-Assessment Test

1. Suppose a graph *G* contains two distinct paths from a vertex *u* to a vertex *v*. Show that *G* has a cycle.

2. Suppose *G* is a finite cycle-free graph with at least one edge. Show that *G*

has at least two vertices of degree 1.

3. Show that a connected graph G with n vertices must have at least $n - 1$ edges.

4. Find the number of connected graphs with four vertices.

5. Suppose G has V vertices and E edges. Let$M$ and m denote, respectively, the maximum and minimum of the degrees of the vertices in G. Show that $m \leq 2E/V \leq M$.

6.16 Answers to check your progress

1. Suppose G is Eulerian and T is a closed Eulerian trail. For any vertex v of G, the trail T enters and leaves v the same number of times without repeating any edge. Hence v has even degree.

Suppose conversely that each vertex of G has even degree.We construct an Eulerian trail.We begin a trail $T1$ at any edge e. We extend $T1$ by adding one edge after the other. If $T1$ is not closed at any step, say, $T1$ begins at u but ends at v _= u, then only an odd number of the edges incident on v appear in $T1$; hence we can extend $T1$ by another edge incident on v. Thus we can continue to extend $T1$ until $T1$ returns to its initial vertex u, i.e., until $T1$ is closed. If $T1$ includes all the edges of G, then $T1$ is our Eulerian trail.

Suppose $T1$ does not include all edges of G. Consider the graph H obtained by deleting all edges of $T1$ from G. H may not be connected, but each vertex of H has even degree since $T1$ contains an even number of the edges incident on any vertex. Since G is connected, there is an edge e of H which has an endpoint u in $T1$. We construct a trail $T2$ in H beginning at u and using e. Since all vertices in H have even degree, we can continue to extend $T2$ in H until $T2$ returns to u as pictured in Fig. 8-41. We can clearly put $T1$ and $T2$ together to form a larger closed trail in G.

We continue this process until all the edges of G are used.We finally obtain an Eulerian trail, and so G is Eulerian.

2. The proof is by induction on n. The theorem is certainly true for the graph with only one vertex and hence no edges. That is, the theorem holds for $n = 1$.We now assume that $n > 1$ and that the theorem holds for graphs with less than n vertices.

(i) *implies* (ii) Suppose G is a tree. Then G is cycle-free, so we only need to show that G has $n-1$ edges, G has a vertex of degree 1. Deleting this vertex and its edge, we obtain a tree T which has $n - 1$ vertices. The theorem holds for T , so T has $n - 2$ edges. Hence G has $n - 1$ edges.

(ii) *implies* (iii) Suppose G is cycle-free and has $n - 1$ edges. We only need show that G is connected. Suppose G is disconnected and has k components, $T1, \ldots, Tk$, which are trees since each is connected and cycle-free. Say Ti has ni vertices. Note $ni < n$. Hence the theorem holds for Ti, so Ti has ni

$- 1$ edges. Thus $n = n1 + n2 + \cdots + nk$ and $n - 1 = (n1 - 1) + (n2 - 1) + \cdots + (nk - 1) = n1 + n2 + \cdots + nk - k = n - k$

Hence $k = 1$. But this contradicts the assumption that G is disconnected and has $k > 1$ components. Hence G is connected.

(iii) *implies* (i) Suppose G is connected and has $n - 1$ edges. We only need to show that G is cycle-free. Suppose G

has a cycle containing an edge e. Deleting e we obtain the graph $H = G - e$ which is also connected. But H has n vertices and $n - 2$ edges, and this contradicts . Thus G is cycle-free and hence is a tree.

3. Suppose the connected map M consists of a single vertex P. Then $V = 1$, $E = 0$, and $R = 1$. Hence $V - E + R = 2$. Otherwise M can be built up from a single vertex by the following two constructions:
(1) Add a new vertex $Q2$ and connect it to an existing vertex $Q1$ by an edge which does not cross any existing
edge
(2) Connect two existing vertices $Q1$ and $Q2$ by an edge e which does not cross any existing edge as in
Neither operation changes the value of $V - E + R$. Hence M has the same value of $V - E + R$ as the map consisting of a single vertex, that is, $V - E + R = 2$. Thus the theorem is proved.
4. $q \geq 3p > 3p - 6$
5. (i) path
(ii) circuit.

6.18 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.
2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.
3 C.L. Liu, Elements of Discrete Mathematics, McGraw-Hili Book Co.
4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 7 **Written by Dr. Vizender Singh**

GRAPH THEORY - II

Structure:

7.0 Learning Objectives

The main objective of this chapter is to familiarize the students with the concept of Eulerian paths and circuits; and planar graphs. The reader will learn about Euler's theorem on the existence of Eulerian paths and circuits, Konigsberg's seven bridges problem. The reader will also study planar graphs and their properties, matrix representation of graphs, directed graphs, strongly connected graphs. Examples are also given to illustrate these topics.

7.1 Introduction

In this chapter we have defined graphs, digraphs and matrix representation of graphs. In Mathematics and Computer Science, Graph theory is the study of graphs, mathematical structures used to model pair wise relations between objects from a certain collection. A graph in this context refers to a collection of vertices or nodes and a collection of edges that connect pairs of vertices. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another. 4 The word graph has atleast two meanings in Mathematics. In Elementary Mathematics, graph refers to a function graph or graph of a function. In Mathematician's terminology, a graph is a collection of points and lines connecting some subset of them. The points of a graph are most commonly known as graph vertices, but may also be called nodes or simply points. Similarly, the lines connecting the vertices of a graph are most commonly known as graph edges, but may also be called arcs or lines.

7.2 Eulerian Paths and Circuits

Definition: A path in a graph G is called an **Euler Path** if it includes every edge exactly once.

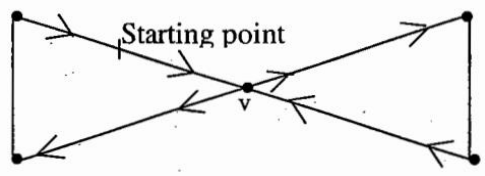
Definition: A circuit in a graph G is called an **Euler Circuit** if it includes every edge exactly once. Thus, an Euler circuit (Eulerian trail) for a graph G is a sequence of adjacent vertices and edges in G that starts and ends at the same vertex, uses every vertex of G at least once, and uses every edge of G exactly once.

Definition: A graph is called **Eulerian graph** if there exists an Euler circuit for that graph.

Theorem 1: If a graph has an Euler circuit, then every vertex of the graph 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.

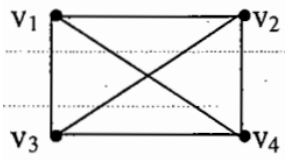


We know that contrapositive of a conditional statement is logically equivalent to statement. Thus the above theorem is equivalent to:

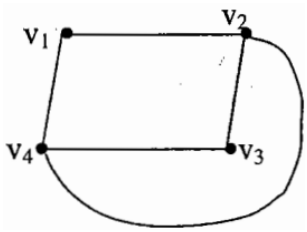
Theorem 2: If a vertex of a graph is not of even degree, then it does not have an Euler circuit. Thus, "If some vertex of a graph has odd degree, then that graph does not have an Euler circuit".

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 an Euler circuit.
In graph (b), we have

$\deg(v_2) = 3$

$\deg(v_4) = 3$

Since there are vertices of odd degree in the given graph, therefore it does not have an Euler circuit.

Remark: The converse of Theorem 1 is not true. There exist graphs in which every vertex has even degree but the Euler circuits do not exist.

For example,

and

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 3: If G is a connected graph and every vertex of G has even degree, then G has an Euler circuit.

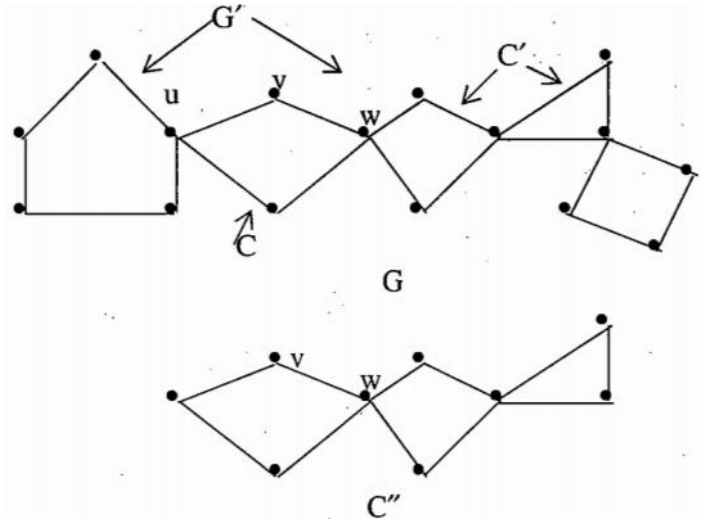
Proof: Let every vertex of a connected graph G has even degree. If G consists of a single vertex, then the trivial walk from v to v is an Euler circuit. So suppose G consists of more than one vertices. We start from any vertex v of G. Since the degree of each vertex of G is even, if we reach each vertex 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 obtain a 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 and vertex of G, then C″ is an Euler circuit. If not, then we again repeat our process. Since the graph is finite, the process must terminate.

The process followed has been described in the graph G shown below:



7.3 Euler's Theorem on the Existence of Eulerian Circuits

Theorem 4: (Euler's Theorem)

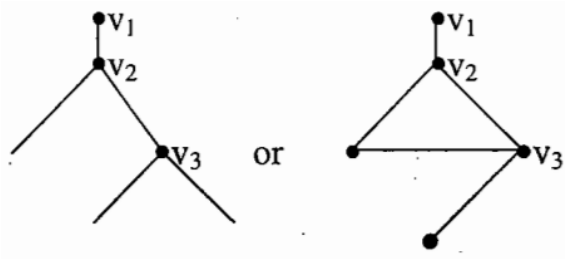
A finite connected graph G has an Euler circuit if and only if every vertex of G has even degree. Thus finite connected graph is Eulerian if and only if each vertex has even degree.

Proof: Theorems 1 and 3 taken together imply the required result.

7.4 Euler's Theorem on the Existence of Eulerian Paths

Theorem 5: If a graph G has more than two vertices of odd degree, then there can be no Euler path in G.

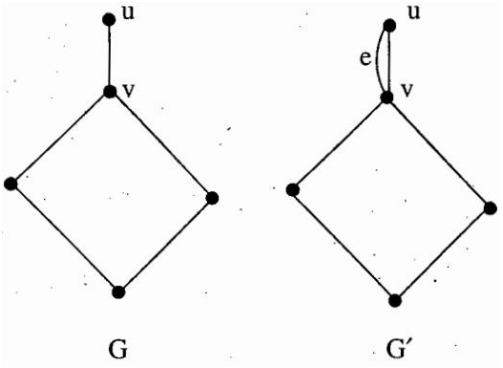
Proof: Let v_1, v_2 and v_3 be vertices of odd degree. Since each of these vertices had odd degree, any possible Euler path must leave (arrive at) each of v_1, v_2, v_3 with no way to return (or leave). One vertex of these three vertices may be the beginning of Euler path and another the end but this 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 6: 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.

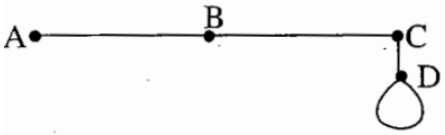


If we add the edge e to G, we get a connected graph G' all of whose vertices have even degree. Hence there will be an Euler circuit in G'. If we omit edge e from Euler circuit, we get an Euler path beginning at u (or v) and ending at v (or u).

Statement of Euler's theorem on the existence of Eulerian paths: A connected graph has an Euler path if and only if it has exactly two vertices of odd degree.

Proof: Theorems 5 and 6 taken together imply the required result.

Example: Has the graph given below an Eulerian path?



Solution: In the given graph,

deg(A) = 1

deg(B) = 2

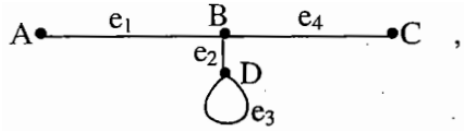
deg(C) = 2

deg(D) = 3

Thus the given connected graph has exactly two vertices of odd degree. Hence, it has an Eulerian path.

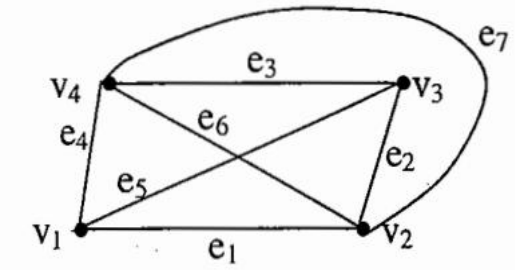
If it starts from A (vertex of odd degree), then it ends at D (vertex of odd degree). If it starts from D (vertex of odd degree), then it ends at A (vertex of odd degree).

But on the other hand, if we have the graph as given below:



Then deg(A) = 1, deg(B) = 3, deg(C) = 1, deg(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

$\deg(v_1) = 3$

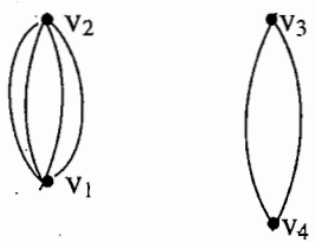
$\deg(v_2) = 4$

$\deg(v_3) = 3$

$\deg(v_4) = 4$

Since this connected graph has vertices with odd degree, it cannot have Euler circuit. But this graph has Euler path, since it has exactly two vertices of odd degree. For example, $v_3\ e_2\ v_2\ e_7\ v_4\ e_6\ v_2\ e_1\ v_1\ e_4\ v_4\ e_3\ v_3\ e_5\ v_1$

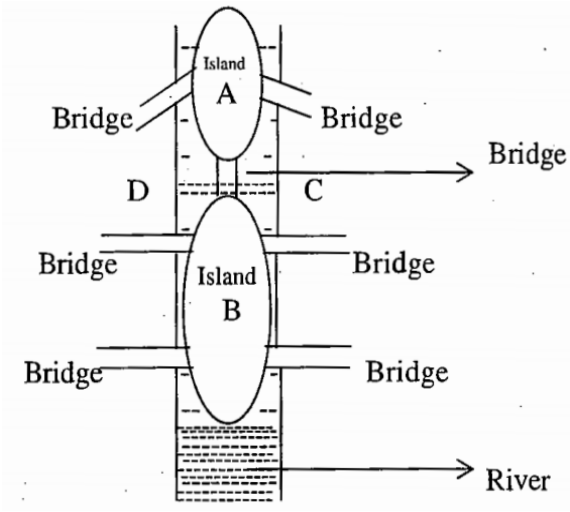
Example: Consider the graph



Here, $\deg(v_1) = 4$, $\deg(v_2) = 4$, $\deg(v_3) = 2$, $\deg(v_4) = 2$. Thus degree of each vertex is even. But the graph is not Eulerian since it is not connected.

7.5 Konigsberg's Seven Bridges Problem

The graph theory began in 1736 when Leonhard Euler solved the problem of seven bridges on Pregel river in the town of Konigsberg in Prussia (now Kaliningrad in Russia). The two islands and seven bridges are shown below:



The people of Konigsberg posed the following question to famous Swiss Mathematician Leonhard Euler:

"Beginning anywhere and ending anywhere, can a person walk through the town of Konigsberg crossing all the seven bridges exactly once?"

Euler showed that such a walk is impossible. He replaced the islands A, B and the two sides (banks) C and D of the river by vertices and the bridges as edges of a graph. We note then that

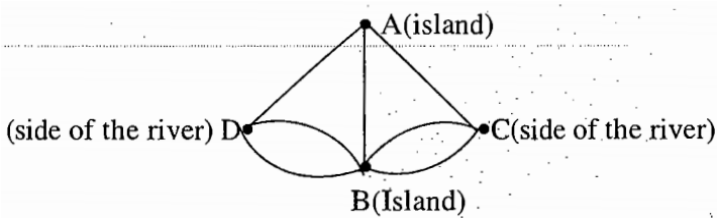
deg(A) = 3

deg(B) = 5

deg(C) = 3

deg(D) = 3

Thus the graph of the problem is



(Euler's graphical representation of seven bridge problem)

The problem then reduces to

"Is there any Euler's path in the above diagram?"

To find the answer, we note that there are more than two vertices having odd degree. Hence there exists no Euler path for this graph.

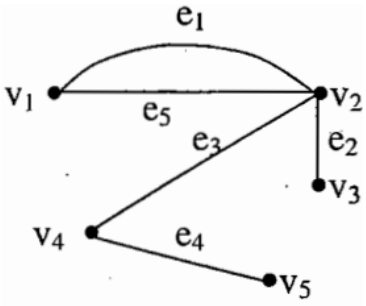
7.6 Bridge or Cut Edge

Definition: An edge in a connected graph is called a **Bridge** or a **Cut Edge** if deleting that edge creates a disconnected graph.

In other words,

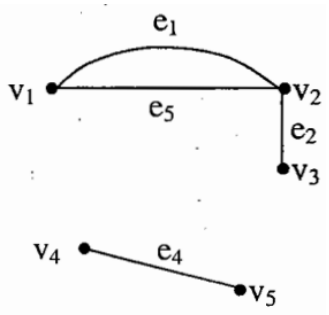
An edge e of a connected graph G is called a bridge (or cut edge) if G – e is disconnected, where G – e is the graph obtained by deleting the edge e.

For example, consider the graph G shown below:



G

In this graph, if we remove the edge e_3, then the graph breaks into two connected components given below:



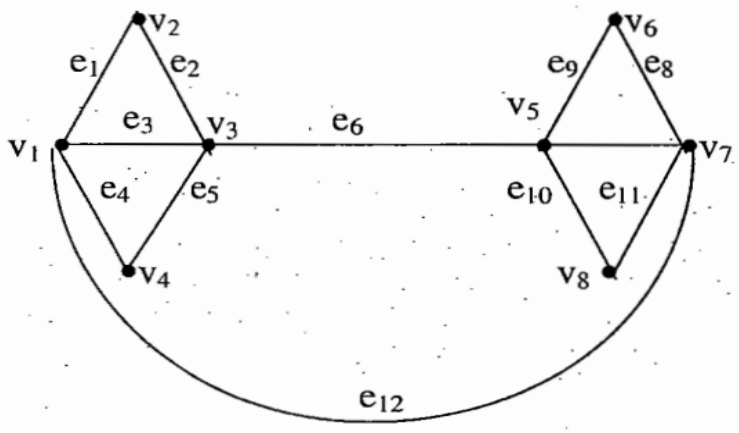
$G - e_3$

Thus the graph $G - e_3$ is disconnected. Hence the edge e_3 is a bridge in the given graph.

7.7 Methods 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

$$\deg(v_2) = \deg(v_4) = \deg(v_6) = \deg(v_8) = 2$$

$$\deg(v_1) = \deg(v_3) = \deg(v_5) = \deg(v_7) = 4$$

Hence all vertices have even degree. Also the given graph is connected. Hence the given graph has an Euler circuit. We start from the vertex v_1 and let C be

$$C : v_1\ v_2\ v_3\ v_1$$

Then C is not an Euler circuit for the given graph but C intersects the rest of the graph at v_1 and v_3. Let C' be

$$C' : v_1\ v_4\ v_3\ v_5\ v_7\ v_6\ v_5\ v_8\ v_7\ v_1$$

(In case we start from v_3, then C' will be $v_3\ v_4\ v_1\ v_7\ v_6\ v_5\ v_7\ v_8\ v_5$)

Path C' into C and obtain

$$C'' : v_1\ v_2\ v_3\ v_1\ v_4\ v_3\ v_5\ v_7\ v_6\ v_5\ v_8\ v_7\ v_1$$

Or we can write

$$C'' : e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7\ e_8\ e_9\ e_{10}\ e_{11}\ e_{12}$$

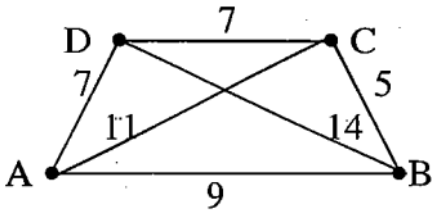
(If we had started from v_2, then C" : $v_1\ v_2\ v_3\ v_4\ v_1\ v_7\ v_6\ v_5\ v_7\ v_8\ v_5\ v_3\ v_1$ or $e_1\ e_2\ e_5\ e_4\ e_{12}\ e_8\ e_9\ e_7\ e_{11}\ e_{10}\ e_6\ e_3$)

In C" all edges are covered exactly once. Also every vertex has been covered at least once. Hence C" is an Euler circuit.

7.8 Weighted Graphs

Definition: A **weighted graph** is a graph for which each edge or each vertex or both is (are) labeled with a numerical value, called its **weight**.

For example, if vertices in a graph denote recreational sites of a town and weights of edges denote the distances in kilometers between the sites, then the graph shown below is a weighted graph.



Definition: The **weight of an edge** (v_i, v_j) is called distance between the vertices v_i and v_j.

Definition: A vertex u is a **nearest neighbour** of vertex v in a graph if u and v are adjacent and no other vertex is joined to v by an edge of lesser weight than (u, v).

For example, in the above example, B is the nearest neighbour of C, whereas A and C are both nearest neighbour of the vertex D. **Thus nearest neighbour of a set of vertices is not unique.**

Definition: A vertex u is a nearest neighbour of a set of vertices $\{v_1, v_2,....,v_n\}$ in a graph if u is adjacent to some member v_i of the set and no other vertex adjacent to member of the set is joined by an edge of lesser weight than (u, v_i).

In the above example if we have set of vertices as {B, D}, C is the neatest neighbour of {B, D} because the edge (C, B) has weight 5 and no other vertex adjacent to {B, D} is linked by an edge of lesser weight than (C, B).

Definition: The length of a path in a graph is the sum of lengths of edges in the path.

Definition: Let G = (V, E) be a graph and let l_{ij} denote the length of edge (v_i, v_j) in G. Then a **shortest path** from v_i to v_k is a path such that the sum of lengths of its edges

$$l_{12} + l_{23} + \ldots\ldots\ldots + l_{k-1,k}$$

is **minimum**, that is, total edge weight is minimum.

7.9 Matrix Representation of Graphs

A graph can be represented inside a computer by using the adjacency matrix or the incidence matrix of the graph.

Definition: Let G be a graph with n ordered vertices $v_1, v_2, \ldots\ldots, v_n$. Then the **adjacency matrix of G** is then n × n matrix $A(G) = (a_{ij})$ over the set of non- negative integers such that

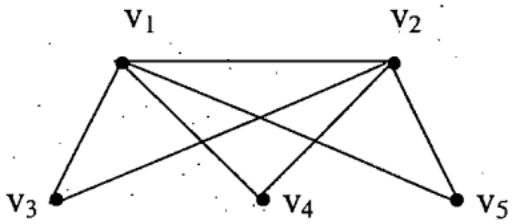
a_{ij} = the number of edges connecting v_i and v_j for all i, j = 1, 2,...,n.

We note that if G has no loop, then there is no edge joining v_i to v_i, i = 1, 2,..., n. Therefore, in this case, all the entries on the main diagonal will be 0.

Further, if G has no parallel edge, then the entries of A(G) are either 0 or 1. It may be noted that adjacent matrix of a graph is symmetric.

Conversely, given a n × n symmetric matrix $A(G) = (a_{ij})$ over the set of non-negative integers, we can associate with it a graph G, whose adjacency matrix is A(G), by letting G have n vertices and joining v_i to vertex v_j by a_{ij} edges.

Example 1: Find the adjacency matrix of the graph shown below:



Solution: The adjacency matrix $A(G) = (a_{ij})$ is the matrix such that

a_{ij} = No. of edges connecting v_i and v_j.

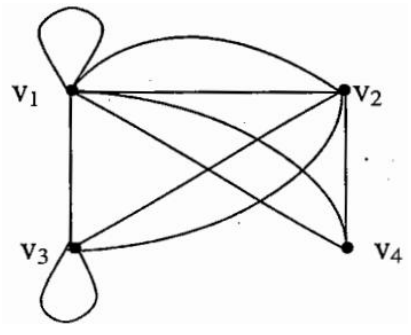
So we have for the given graph

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Example 2: Find the graph that have the following adjacency matrix

$$\begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 0 & 2 & 1 \\ 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

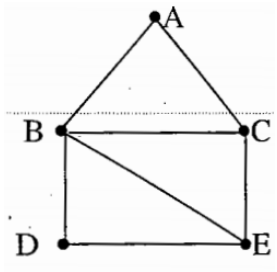
Solution: We note that there is a loop at v_1 and a loop at v_3. There are parallel edges between v_1, v_2; v_1, v_4; v_2, v_1; v_2, v_3; v_3, v_2; v_4, v_1. Thus the graph is



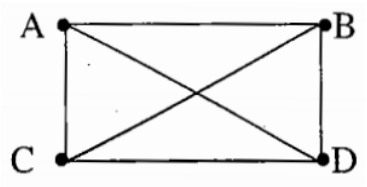
7.10 Planar Graphs

Definition: A graph which can be drawn in the plane so that its edges do not cross is said to be **planar.**

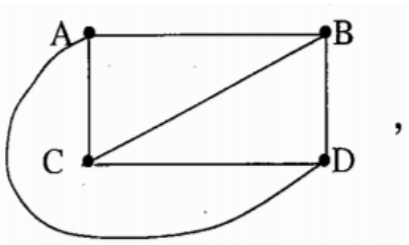
For example, the graph shown below is planar:



Also the complete graph K_4 shown, below is planar:

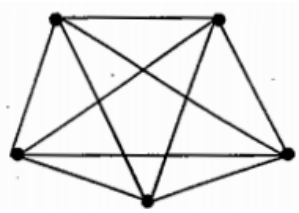


In fact, it can be redrawn as



so that no edges cross.

But the complete map K_5 is not planar because in this case, the edges cross each others.

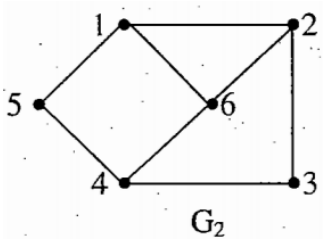


Definition: An area of the plane that is bounded by edges of the planar graph and is not further subdivided into subareas is called a **region** or **face** of a planar graph.

A face is characterized by the cycle that forms its boundary.

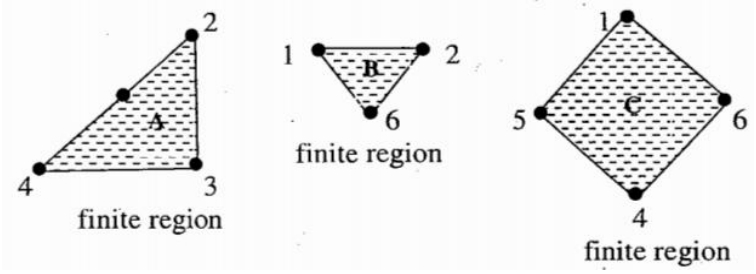
Definition: A region is said to be **finite** if its area is finite and **infinite** if its area is infinite. Clearly a planar graph has exactly one infinite region.

For example, consider the graph:

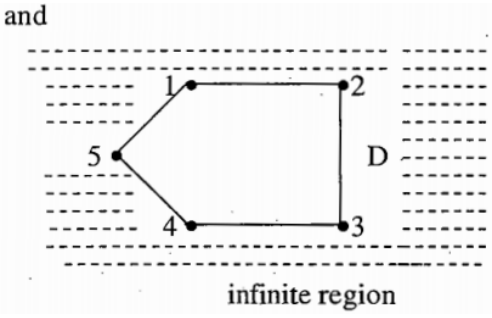


G_2

In graph G_2, there are four region A, B, C, D as shown below:



and



Definition: Let f be a face (region) in a planar graph. The length of the cycle (or closed walk) which borders f is called the **degree of the region f**. It is denoted by deg(f).

In a planar graph we note that each edge either borders two regions or is contained in a region and will occur twice in any walk along the border of the region. Thus we have

Theorem: The sum of the degrees of the regions of a map is equal to twice the number of edges.

For example, in the graph G_2, discussed above, we have

deg(A) = 4, deg(B) = 3, deg(C) = 4, deg (D) = 5

The sum of degrees of all regions = 4 + 3 + 4 + 5 = 16

No. of edges in G_2 = 8

Hence, "sum of degrees of region is twice the number of edges".

7.11 Euler's Formula for Connected Planar Graphs

Theorem: If G is a connected planar graph with e edges, v vertices and r regions, then

v – e + r = 2

Proof: We shall use induction on the number of edges. Suppose that e = 0. Then the graph G consists of a single vertex, say P. Thus G is as shown below:

• P

In this case, we have

e = 0, v = 1, r = 1

Thus

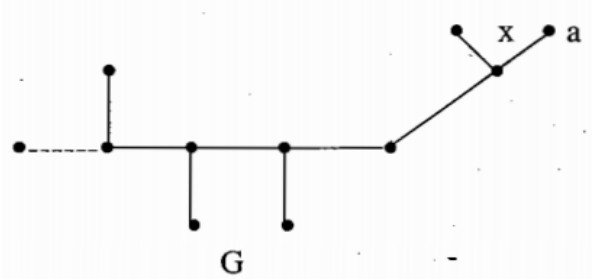
v – e + r = 1 – 0 + 1 = 2

Hence, the formula holds in this case.

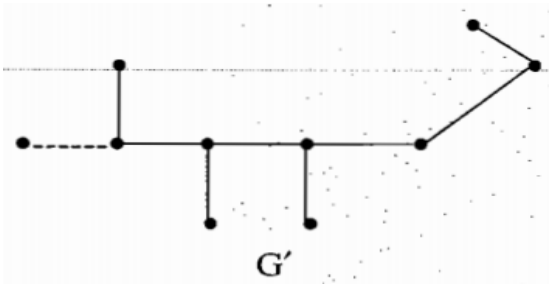
Suppose that e = 1. Then the graph G is one of the two graphs shown below:

We see that, in either case, the formula holds.

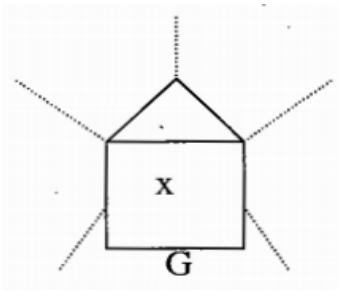
Suppose that the formula holds for connected planar graph with n edges. We shall prove that this holds for graph with n + 1 edges. So, let G be the graph with n + 1 edges. Suppose first that G contains no cycles. Choose a vertex v_1 and trace a path starting at v_1. Ultimately, we will reach a vertex "a" with degree 1, that we cannot leave.



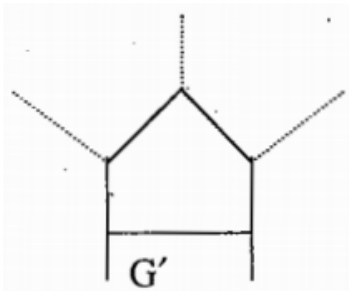
We delete "a" and the edge x incident on "a" from the graph G. The resulting graph G' has n edges and so by induction hypothesis, the formula holds for G'. Since G has one more edge than G', one more vertex than G' and the same number of faces as G', it follows that the formula v – e + r = 2 holds also for G.



Now suppose that G contains a cycle. Let x be an edge in a cycle.



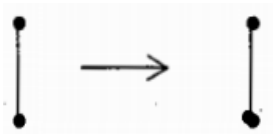
Now the edge x is part of a boundary for two faces. We delete the edge x but no vertices to obtain the graph G'.



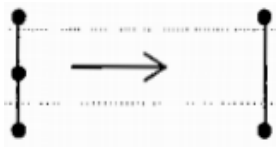
Thus G' has n edges and so by induction hypothesis the formula holds. Since G has one more face (region) than G', one more edge than G' and the same number of vertices as G', it follows that the formula v – e + r = 2 also holds for G. Hence, by Mathematical Induction, the theorem is true.

Remarks: Planarity of a graph is not affected if

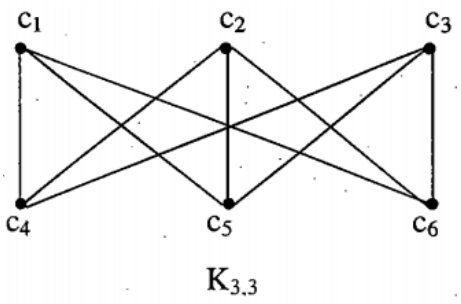
(i) an edge is divided into two edges by the insertion of new vertex of degree 2.



(ii) two edges that are incident with a vertex of degree 2 are combined as a single edge by the removal of that vertex.



Example: Show that the graph $K_{3,3}$ given below, is not planar.



Solution: Suppose that $K_{3,3}$ is planar. Since every cycle in $K_{3,3}$ has at least four edges, each face (region) is bounded by at least four edges. Thus the number of edges that bound regions is at least 4r. Also, in a planar graph each edge belongs to atmost two bounding cycles. Therefore,

$2e \geq 4r$ (sums of degrees of region is equal to twice the number of edges)

But, by Euler's formula for planar graph,

$$r = e - v + 2$$

Hence

$$2e \geq 4 (e - v + 2) \qquad (1)$$

In case of $K_{3,3}$ we have

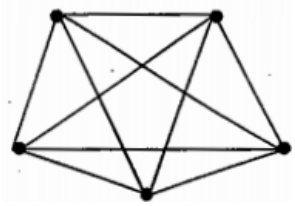
$$e = 9, v = 6$$

and so (1) yields

$$18 \geq 4 (9 - 6 + 2) = 20 ,$$

which is a contradiction. Therefore, the graph $K_{3,3}$ is not planar.

Remark: By an argument similar to the above example, we can show that the graph K_5 (given below) is not planar.



(non- planar graph K_5)

We observe that if a graph contains $K_{3,3}$ or K_5 as a subgraph, then it cannot be planar.

7.12 Kuratowski Graphs

The complete graph K_5 and the complete bipartite graph $K_{3,3}$ are called the Kuratowski graphs.

7.13 Directed Graphs

Definition: A **directed graph or digraph** consists of two finite sets:

(i) A set V of vertices (or nodes or points)

(ii) A set E of directed edges (or arcs), where each edge is associated with an ordered pair (v, w) of vertices called its endpoints. If edge e is associated with the ordered pair (v, w), then edge e is said to be **directed edge from v to w**.

The directed edges are indicated by arrows.

We say that edge e = (v, w) is incident from v and is incident into w.

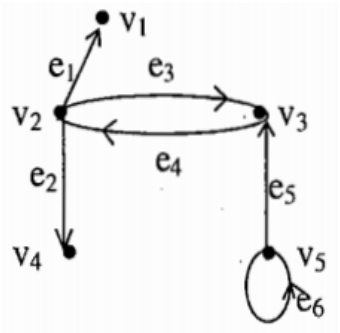
The vertex v is called **initial vertex** and the vertex w is called the **terminal vertex** of the directed edge (v, w).

7.14 Indegree and Outdegree of a Vertex

Definition: Let G be a directed graph. The **outdegree of a vertex v of G** is the number of edges beginning at v. It is denoted by outdeg(v).

Definition: Let G be a directed graph. The **indegree of a vertex v of G** is the number of edges ending at v. It is denoted by indeg(v).

Example: Consider the directed graph shown below:



Here edge e_1 is (v_2, v_1) whereas e_6 is denoted by (v_5, v_5) and is called a loop. The indegree of v_2 is 1, outdegree of v_2 is 3.

Definition: A vertex with 0 indegree is called a **source**, whereas a vertex with 0 outdegree is called a **sink**. For instance, in the above example, the vertex v_1 is a sink.

Definition: If the edges and/or vertices of a directed graph G are labeled with some type of data, then G is called a **Labeled Directed Graph.**

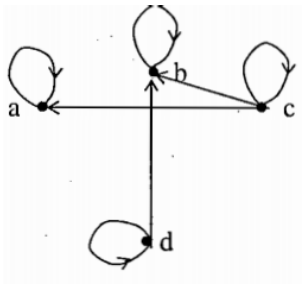
7.15 Adjacency Matrix of Directed Graph

Definition: Let G be a **directed graph** with ordered vertices $v_1, v_2, \ldots\ldots, v_n$. **The adjacency matrix of G** is the matrix $A = (a_{ij})$ over the set of non-negative integers such that

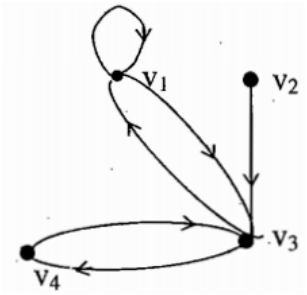
a_{ij} = the number of arrows from v_i to v_j , i, j = 1, 2, ...,n.

Example 1: Find the adjacency matrices for the graphs given below:

(i)



(ii)



Solution: (i) The edges in the directed graph are (a, a), (b, b), (c, c), (d, d), (c, a), (c, b) and (d, b). Therefore the adjacency matrix $A = (a_{ij})$ is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

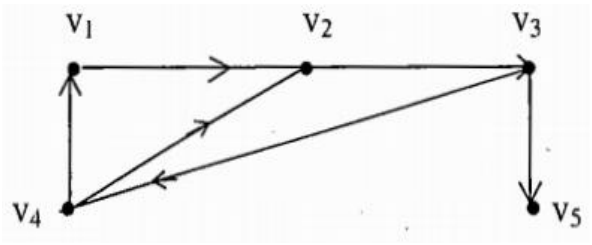
(ii) The edges in the graph are (v_2, v_3), (v_1, v_1), (v_1, v_3), (v_3, v_1), (v_3, v_4), (v_4, v_3). Hence the adjacency matrix is

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Example 2: Find the directed graph represented by the adjacency matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

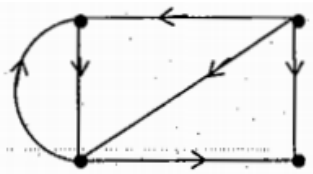
Solution: We observe that $a_{12} = 1$, $a_{23} = 1$, $a_{34} = 1$, $a_{35} = 1$, $a_{41} = 1$, $a_{42} = 1$. Hence the digraph is as shown below:



7.16 Simple Directed Graph

Definition: In a directed graph, if there is no more than one directed edge in a particular direction between a pair of vertices, then it is called **simple directed graph.**

For example, the graph



is a simple directed graph.

A directed graph which is not simple is called **directed multigraph.**

7.17 Check Your Progress

1. The following statements are equivalent for a connected graph G :

(i) G is Eulerian

(ii) Every point of G has even degree

(iii)The set of lines of G be partitioned into cycles.

2. A connected graph G has an Eulerian trail if and only if it has at most two odd vertices. i.e., it has either no vertices of odd degree or exactly two vertices of odd degree.

3. Show that a connected graph with exactly two odd vertices is a unicursal graph.

4. If G is a graph in which the degree of each vertex is at least 2, then G contains a cycle.

5. Let G be a simple graph with n vertices and let u and v be an edge. Then G is hamiltonian if and only if G + uv is hamiltonian.

7.18 Summary

In this chapter we have discussed Eulerian paths and circuits, Euler's theorem on the existence of Eulerian paths and circuits, Konigsberg's seven bridges problem. We have also discussed planar graphs and their properties, matrix representation of graphs, directed graphs, strongly connected graphs. Examples have given to illustrate these topics.

7.19 Keywords

Eulerian paths and circuits, planar graphs, adjacency matrix, directed graphs

7.20 Self-Assessment Test

1. How many vertices do the following graphs have if they contain

(i) 16 edges and all vertices of degree 2

(ii) 21 edges, 3 vertices of degree 4 and others each of degree 3.

2. Suppose a graph has vertices of degree 0, 2, 2, 3 and 9. How many edges does the graph have ?

3. Draw

(i) a simple graph,

(ii) a non-simple graph with no loops,

(iii) a non-simple graph with no multiple edges, each with five vertices and eight Edges.

4. In a graph G, let P1 and P2 be two different paths between two given vertices. Prove that G has a circuit in it.

5. Prove that a connected graph of order n contains exactly one circuit if and only if its size is also n.

7.21 Answers to check your progress

1. (i) implies (ii) Let T be an Eulerian trail in G. Each occurrence of a given point in T contributes 2 to the degree of that point, and since each line of G appears exactly once in T, every point must have even degree.

(ii) implies (iii) Since G is connected and non trivial, every point has degree at least 2, so G contains a cycle Z. The removal of the lines of Z results in a spanning subgraph G1 in which every point still has even degree. If G1 has no lines, then (iii) already holds ; otherwise, repetition of the argument applied to G1 results in a graph G2 in which again all points are even, etc. When a totally disconnected graph Gn is obtained, we have a partition of the lines of G into n cycles.

(iii) implies (i) Let Z1 be one of the cycles of this partition. If G consists only of this cycle, then G is obviously Eulerian. Otherwise, there is another cycle Z2 with a point v in common with Z1. The walk beginning at v and consisting of the cycles Z1 and Z2 in succession is a closed trail containing the lines of these two cycles. By continuing this process, we can construct a closed trail containing all lines of G. Hence G is Eulerian.

2. Suppose G has an Eulerian trail which is not closed. Since each vertex in the middle of the trail is associated with two edges and since there is only one edge associated with each end vertex of the trail, these end vertices must be odd and the other vertices must be even. Conversely, suppose that G is connected with atmost two odd vertices. If G has no odd vertices then G is Euler and so has Eulerian trail. The leaves us to treat the case when G has two odd vertices (G cannot have just one odd vertex since in any graph there is an even number of vertices with odd degree).

3. Suppose A and B be the only two odd vertices in a connected graph G. Join these vertices by an edge e. Then A and B become even vertices. Since all other vertices in G are of even degree, the graph G ∪ e is an Eulerian graph. Therefore, it has an Euler line which must include. The open walk got by deleting e from this Euler line is a semi-Euler line in G. Hence G is a unicursal graph.

4. If G has any loops or multiple edges, the result is trivial. Suppose that G is a simple graph. Let v be any vertex of G. We construct a walk $v \rightarrow v1 \rightarrow v2 \rightarrow$ ..... inductively by choosing v1 to be any vertex adjacent to v and for each $i > 1$. Choosing $vi + 1$ to be any vertex adjacent to vi except $vi - 1$, the existence of such a vertex is guaranteed by our hypothesis. Since G has only finitely many vertices, we must eventually choose a vertex that has been chosen before. If vk is the first such vertex, then that part of the walk lying between the two occurrences of vk is the required cycle.

5. Suppose G is hamiltonian. Then the super graph G + uv must also be hamiltonian. Conversely, suppose taht G + uv is hamiltonian. Then if G is not hamiltonian. i.e., if G is a graph with $p \geq 3$ vertices such that for all non adjacent vertices u and v, deg u + deg $v \geq p$. We obtain the inequality deg u + deg $v < n$. However by hypothesis, deg u + deg $v \geq n$. Hence G must be hamiltonian. This completes the proof.

7.22 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.

2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.

3 C.L. Liu, Elements of Discrete Mathematics, McGraw-

Hili Book Co.

4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

MAL-637: M. Sc. Mathematics (Advanced Discrete Mathematics)

Lesson No. 8 **Written by Dr. Vizender Singh**

TREES

Structure:

8.0 Learning Objectives

The main objective of this chapter is to gain knowledge about trees. The reader will know about definition and properties of trees, directed trees, rooted trees, binary trees, spanning tree of a graph, minimal spanning tree. Further, the reader will study Prim algorithm and Kruskal's algorithm for finding minimal spanning tree for a weighted graph. The reader will also learn different methods of searching a tree. Examples are also given to illustrate these topics.

8.1 Introduction

In the previous chapters, we studied some basic definitions of graph theory and matrices associated with graph. We have also discussed Euler paths and circuits, planar graphs and their properties. In the present chapter we will know about trees, binary trees, weighted graphs, minimal spanning tree and tree searching. In the field of discrete mathematics the graph theory has the wonderful and remarkable application not only in theoretical mathematics but also in the real time applications. In a simple word the graph theory is the mathematics object study which is termed as graphs. The graph is defined as follows: It's a triplet contains node or points that are vertex V of any graph G with line edge E which is study of the relationship of vertices and edge [12]. The conventional way of presenting the graph is by considering the dot for the individual vertex and connecting the above dots by the drawing the line between them on two vertices and the edge. The Seven Bridges of Konigsberg is the research article done by the Leonhard Euler in the year of 1736 is considered as the very first research work in the field of graph theory. Such a unsolvable problems made the growth of Graph theory and use the surprising application of the same.

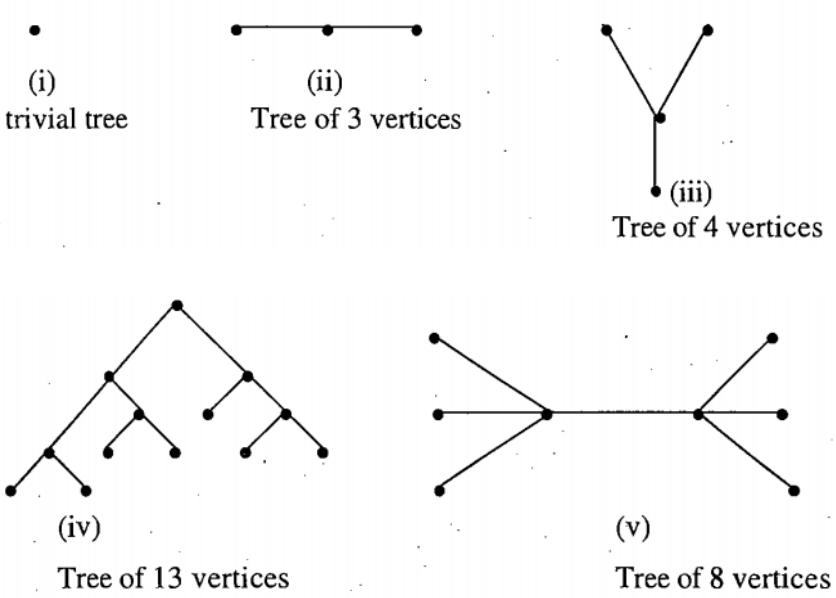
8.2 Some Basic Definitions

8.2.1 Tree

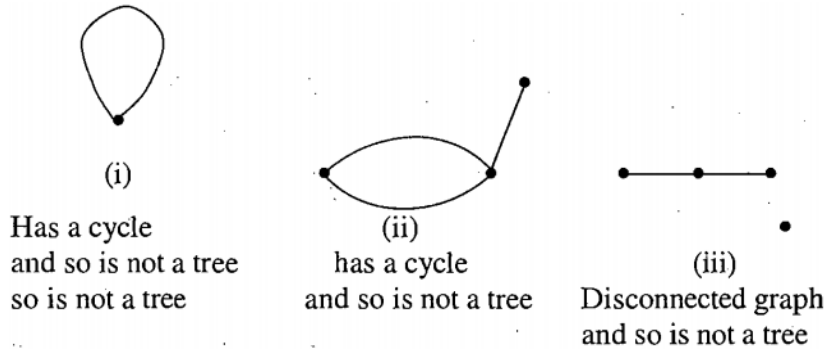
Definition: A graph is said to be a **Tree** if it is a connected acyclic graph.

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.



But the graphs shown below are not trees:



8.2.2 Forest

Definition: A collection of disjoint trees is called a **forest**. Thus a graph is a forest if and only if it is circuit free.

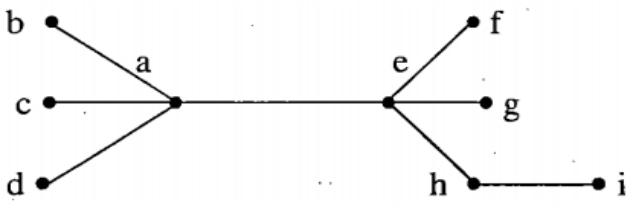
8.2.3 Leaf or Terminal Node

Definition: A vertex of degree 1 in a tree is called a **leaf** or a **terminal node** or a **terminal vertex.**

8.2.4 Branch Node or Internal Node

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:



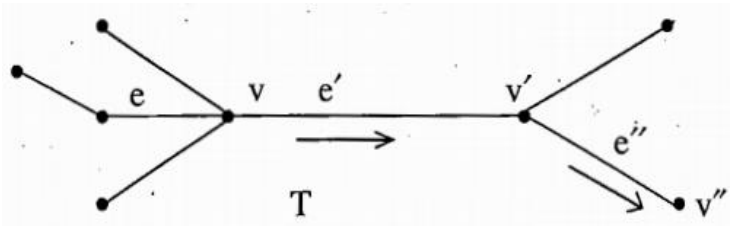
In this tree the vertices b, c, d, f, g, and i are leaves whereas the vertices a, e, h are branch nodes.

8.3 Characterization of Trees

We have the following interesting characterization of trees:

Lemma 1: A tree that has more than one vertex has at least one vertex of degree 1.

Proof: Let T be a particular but arbitrary chosen tree having more than one vertex.



1. Choose a vertex v of T. Since T is connected and has at least two vertices, so v is not isolated and there is an edge e incident on v.

2. If deg (v) > 1, there is an edge e' ≠ e because there are, in such a case, at least two edges incident on v. Let v' be the vertex at the other end of e'. This is possible because e' is not a loop by the definition of a tree.

3. If deg (v') > 1, then there are at least two edges incident on v'. Let e'' be the other edge different from e' and v'' be the vertex at other end of e''. This is again possible because T is acyclic.

4. If deg (v'') > 1, then repeat the above process. Since the number of vertices of a tree is finite and T is circuit free, the process must terminate and we shall arrive at a vertex of degree l.

Remark: In the proof of the above lemma, after finding a vertex of degree 1, if we return to v and move along a path outward from v starting with e, we shall reach to a vertex of degree 1 again. Thus it follows that **"Any tree that has more than one vertex has at least two vertices of degree 1".**

Lemma 2: There is a unique path between every two vertices in a tree.

Proof: Suppose on the contrary that there are more than one path between any two vertices in a given tree T. Then T has a cycle which contradicts the definition of a tree because T is acyclic. Hence the lemma is proved.

Lemma 3: The number of vertices is one more than the number of edges in a tree.

Or

For any positive integer n, a tree with n vertices has n– 1 edges.

Proof: We shall prove the lemma by mathematical induction.

Let T be a tree with one vertex. Then T has no edges, i.e., T has 0 edge. But 0 = 1 – 1. Hence the lemma is true for n = 1.

Suppose that the lemma is true for k > 1. We shall show that it is then true for k + 1 also. Since the lemma is true for k, the tree has k vertices and k–1 edges. Let T be a tree with k + 1 vertices. Since k is +ve, k+1 ≥ 2 and so T has more than one vertex. Hence, by Lemma 1, T has a vertex v of degree 1. Also there is another vertex w and so there is an edge e connecting v and w. Define a subgraph T' of T so that

$$V(T') = V(T) - \{v\}$$

$$E(T') = E(T) - \{ e\}$$

Then number of vertices in T' = (k + l) – 1 = k and since T is circuit free and T' has been obtained on removing one edge and one vertex, it follows that T' is acyclic. Also T' is connected. Hence T' is a tree having k vertices and therefore by induction hypothesis, the number of edges in T' is k–1. But then

No. of edges in T = number of edges in T' + 1 = k – 1+1 = k

Thus the Lemma is true for tree having k + 1 vertices. Hence the lemma is true by mathematical induction.

Corollary 1: Let C(G) denote the number of components of a graph. Then a forest G on n vertices has n – C(G) edges.

Proof: Apply Lemma 3 to each component of the forest G.

Corollary 2: Any graph G on n vertices has at least n – C(G) edges.

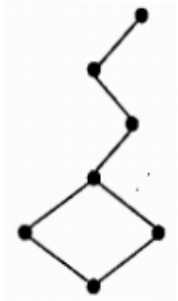
Proof: If G has cycle-edges, remove them one at a time until the resulting graph G* is acyclic. Then G* has n – C(G*) edges by corollary 1. Since we have removed only circuit, C(G*) = C(G). Thus G* has n – C(G) edges. Hence, the graph G has at least n – C(G) edges.

Lemma 4: A graph in which there is a unique path between every pair of vertices is a tree. (This lemma is converse of Lemma 2).

Proof: Since there is a path between every pair of vertices, therefore the graph is connected. Since a path between every pair of vertices is unique, there does not exist any circuit because existence of circuit implies existence of distinct paths between pair of vertices. Thus the graph is connected and acyclic and so is a tree.

Lemma 5: (Converse of Lemma 3) A connected graph G with e = v – 1 is a tree.

Proof: The given graph is connected and e = v – 1. To prove that G is a tree, it is sufficient to show that G is acyclic. Suppose on the contrary that G has a cycle. Let m be the number of vertices in this cycle. Also, we know that number of edges in a cycle is equal to number of vertices in that cycle. Therefore, number of edges in the present case is m. Since the graph is connected, every vertex of the graph which is not in cycle must be connected to the vertices in the cycle.

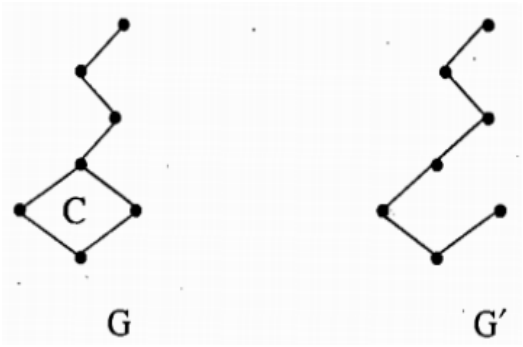


Now each edge of the graph that is not in the cycle can connect only one vertex to the vertices in the cycle. There are v – m vertices that are not in the cycle. So the graph must contain at least v – m edges that are not in the cycle. Thus we have

$$e \geq v - m + m = v,$$

which is a contradiction to our hypothesis. Hence there is no cycle and so the graph in a tree.

Second proof of Lemma 5: We shall show that a connected graph with v vertices and v – 1 edges is a tree. It is sufficient to show that G is acyclic. Suppose on the contrary that G is not circuit free and has a non trivial circuit C. If we remove one edge of C from the graph G, we obtain a graph G' which is connected.



If G' still has a nontrivial circuit, we repeat the above process and remove one edge of that circuit obtaining a new connected graph. Continuing this process, we obtain a connected graph G* which is circuit free. Hence G* is a tree. Since no vertex has been removed, the tree G* has v vertices. Therefore, by Lemma 3, G* has v–1 edges. But at least one edge of G has been removed to form G*. This means that G* has not more than v – 1 – 1 = v – 2 edges. Thus we arrive at a contradiction. Hence our supposition is wrong and G has no cycle. Therefore G is connected and cycle free and so is a tree.

Lemma 6: A graph G with e = v – 1, that has no circuit is a tree.

Proof: It is sufficient to show that G is connected. Suppose G is not connected and let G', G"..... be connected component of G. Since each of G', G", ... is connected and has no cycle, they all are tree. Therefore, by Lemma 3,

e' = v' – 1

e" = v" – 1

---

where e', e", ... are the number of edges and v', v",..... are the number of vertices in G', G", ... respectively. We have, on adding

e' + e" + ... = (v' – 1) + (v" – 1) +.....

Since e = e' + e" +.....

v = v' + v" +.....

we have

e < v – 1

which contradicts our hypotheses. Hence G is connected. So G is connected and acyclic and is therefore a tree.

Example: 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.

Theorem: Let G be a connected graph with n vertices. Then G is a tree if and only if every edge of G is a bridge (cut edge).

(This theorem asserts that every edge in a tree is a bridge).

Proof: Let G be a tree. Then it is connected and has n –1 edges. Let e be an arbitrary edge of G. Since G – e has n – 2 edges, and also we know that a graph G with n vertices has at least n – C(G) edges, it follows that $n - 2 \geq n - C(G - e)$. Thus G – e has at least two components. Thus removal of the edge e created more components than in the graph G. Hence e is a cut edge. This proves that every edge in a tree is a bridge.

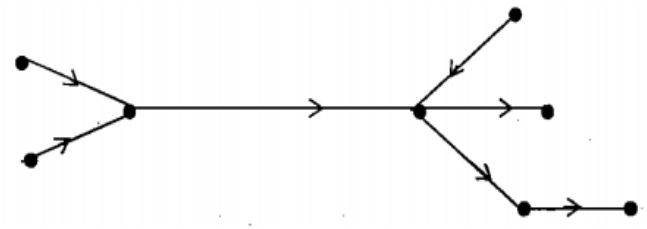
Conversely, suppose that G is connected and every edge of G is a bridge. We have to show that G is a tree. To prove it, we have only to show that G is circuit free. Suppose on the contrary that there exists a cycle between two points x and y in G.

Then any edge on this cycle is not a cut edge which contradicts the fact that every edge of G is a bridge (cut edge). Hence G has no cycle. Thus G is connected and acyclic and so is a tree.

8.4 Directed Tree

Definition: A directed graph is said to be a directed tree if it becomes a tree when the direction of edges are ignored.

For example, the graph shown below is a directed tree.



8.5 Rooted 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 v_0 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.

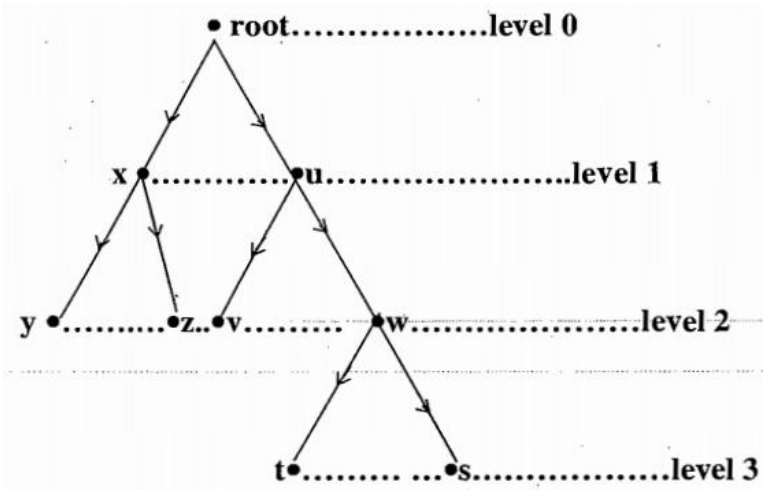
Definition: Two vertices in a rooted tree are said to be **siblings** if they are both children of same parent.

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:



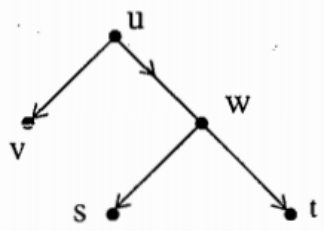
Here y is a child of x; x is the parent of y and z. Thus y and z are siblings. The descendents of u are v, w, t and s. Levels of vertices are shown in the figure. The height of this rooted tree is 3.

8.6 Subtree of a Tree

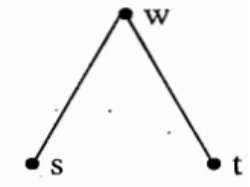
Definition: Let u be a branch node in the tree T = (V, E). Then the subgraph T' (V', E') of T such that the vertices set V' contains u and all of its descendents and E' contains all the edges in all directed paths emerging from u is called a **subtree** with u as the root.

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,



whereas the figure shown below is a subtree of the branch node u .



Example: 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,

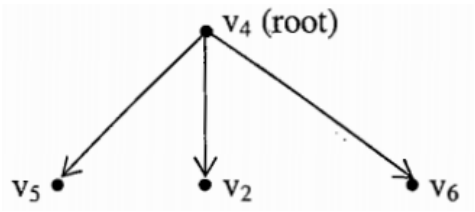
Outgoing degree of $v_2 = 2$

Outgoing degree of $v_4 = 3$

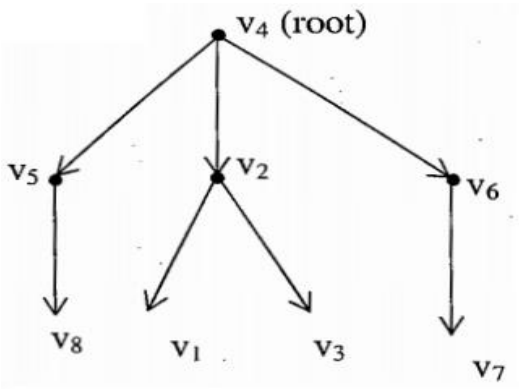
Outgoing degree of $v_5 = 1$

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.

8.7 Ordered Tree

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**.

8.8 Binary Tree

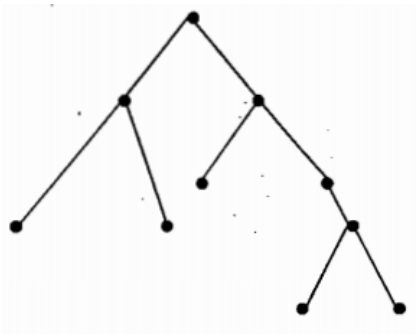
Definition: An ordered tree in which every branch node has atmost n offspring's is called a **n-ary tree** (or **n- tree**).

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.

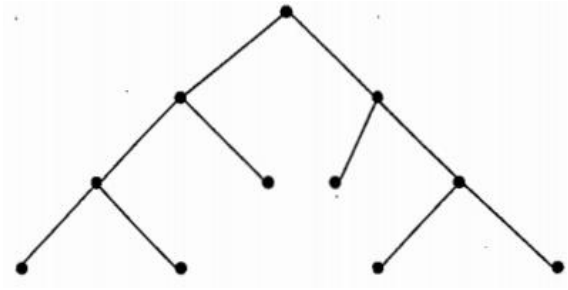
Definition: An ordered tree in which every branch node has almost 2 offsprings is called a **binary tree** (or **2 - tree**).

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.

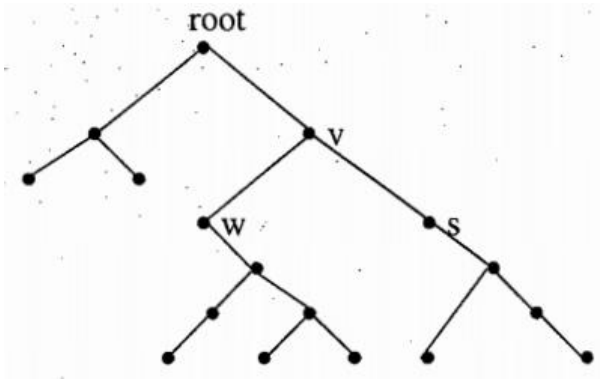


8.9 Left Subtree and Right Subtree of a Binary Tree

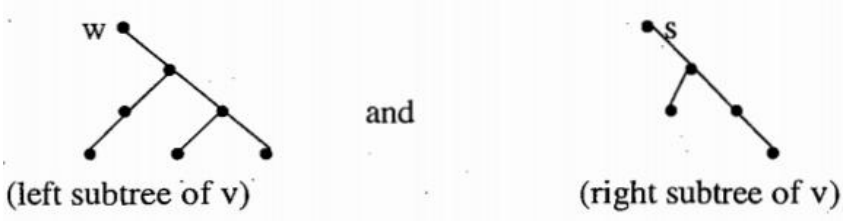
Definition: Let v be a branch node of a binary tree T. The **left subtree** of v is the binary tree whose root is the left child of v, whose vertices consists of the left child of v and all its descendents and whose edges consist of all those edges of T that connects the vertices of the left subtree together.

The **right subtree** can be defined analogously.

For example, the left subtree and the right subtree of v in the tree (shown below):

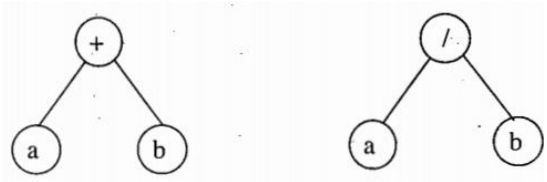


are respectively

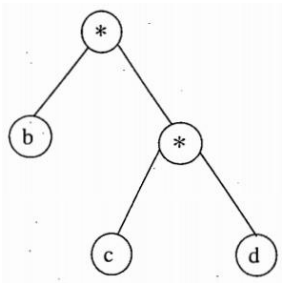


8.10 Representation of Arithmetic/Algebraic Expressions by Binary Trees

Binary trees are used in computer science to represent algebraic expressions involving parentheses. For example, the binary trees



and



represent the expressions

a + b , a/b

and

b * (c *d)

respectively.

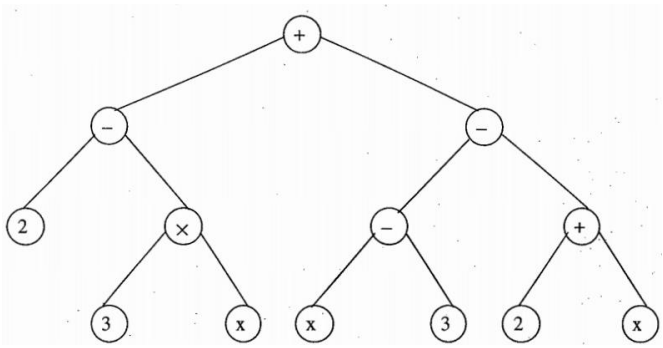
Thus, the central operator acts as root of the tree.

Example 1: Draw a binary tree to represent

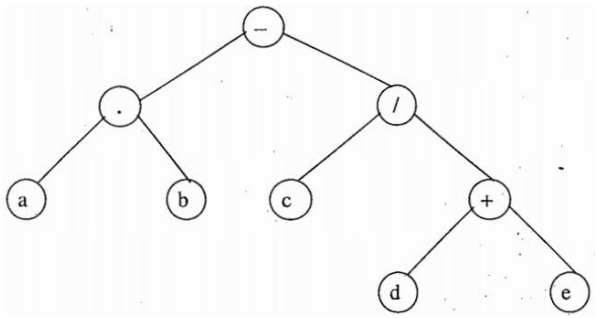
(i) (2 – (3 × x) + ((x – 3) – (2 + x))

(ii) a.b – (c/(d+e)).

Solution: (i) In this expression, + is the central operator. Therefore, the root of tree is +. The binary tree is



(ii) Here the central operator is –. Therefore, it is the root of the tree. We have the following binary tree to represent this expression.



8.11 Spanning Tree of a Graph

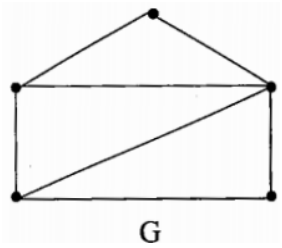
Definition: Let G be a graph, then a subgraph of G which is a tree is called **tree of the graph.**

Definition: A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

Or

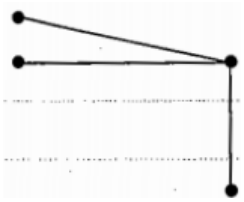
"A **spanning tree** for a graph G is a spanning subgroup of G which is a tree".

Example: Determine a tree and a spanning tree for the connected graph given below:

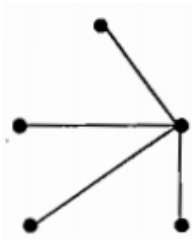


G

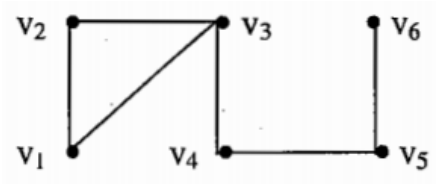
Solution: The given graph G contains circuits and we know that removal of the circuits gives a tree. So, we note that the figure below is a tree.



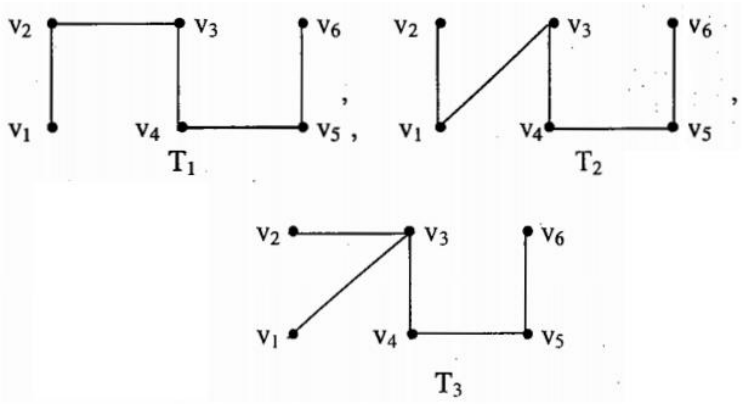
And the figure below is a spanning tree of the graph G.



Example: Find all spanning trees for the graph G shown below:



Solution: The given graph G has a circuit $v_1\ v_2\ v_3\ v_1$. We know that removal of any edge of the circuit gives a tree. So the spanning trees of G are



Remark: We know that a tree with n vertices has exactly n – 1 edges. Therefore if G is a connected graph with n vertices and m edges, then a spanning tree of G must have n – 1 edges. Hence the number of edges that must be removed before a spanning tree is obtained must be

$$m - (n - 1) = m - n + 1.$$

For Illustration, in the above example, n = 6, m = 6, so, we had to remove one edge to obtain a spanning tree.

Theorem: A graph G has a spanning tree if and only if G is connected.

Proof: Suppose first that a graph G has a spanning tree T. If v and w are vertices of G, then they are also vertices in T and since T is a tree there is a path from v to w in T. This path is also a path in G. Thus every two vertices are connected in G. Hence G is connected.

Conversely, suppose that G is connected. If G is acyclic, then G is its own spanning tree and we are done. So suppose that G contains a cycle C_1. If we remove an edge from the cycle, then the subgraph of G so obtained is also connected. If it is acyclic, then it is a spanning tree and we are done. If not, it has at least one circuit, say C_2. Removing one edge from C_2, we get a subgraph of

G which is connected. Continuing in this way, we obtain a connected circuit free subgraph T of G. Since T contains all vertices of G, it is a spanning tree of G.

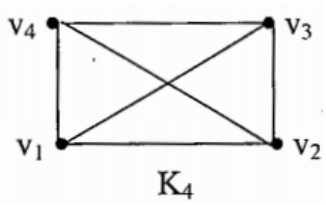
Theorem: Cayley's Formula

The number of spanning trees of the complete graph K_n, $n \geq 2$ is n^{n-2}.

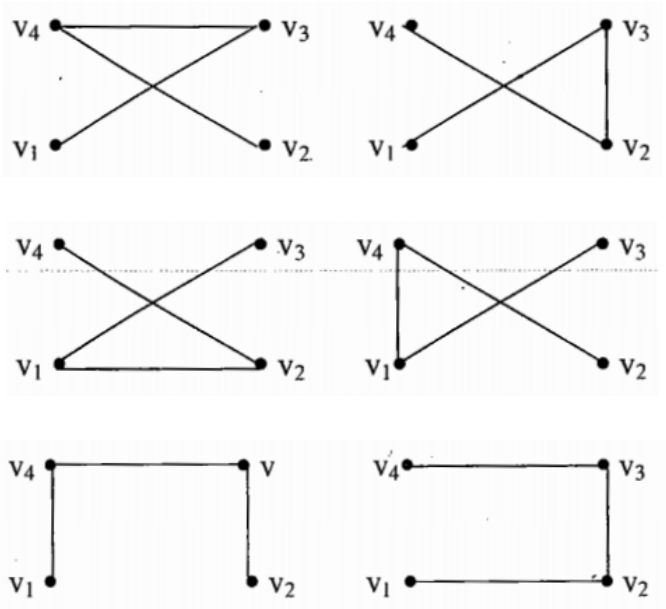
(Proof of this formula is out of scope of this book)

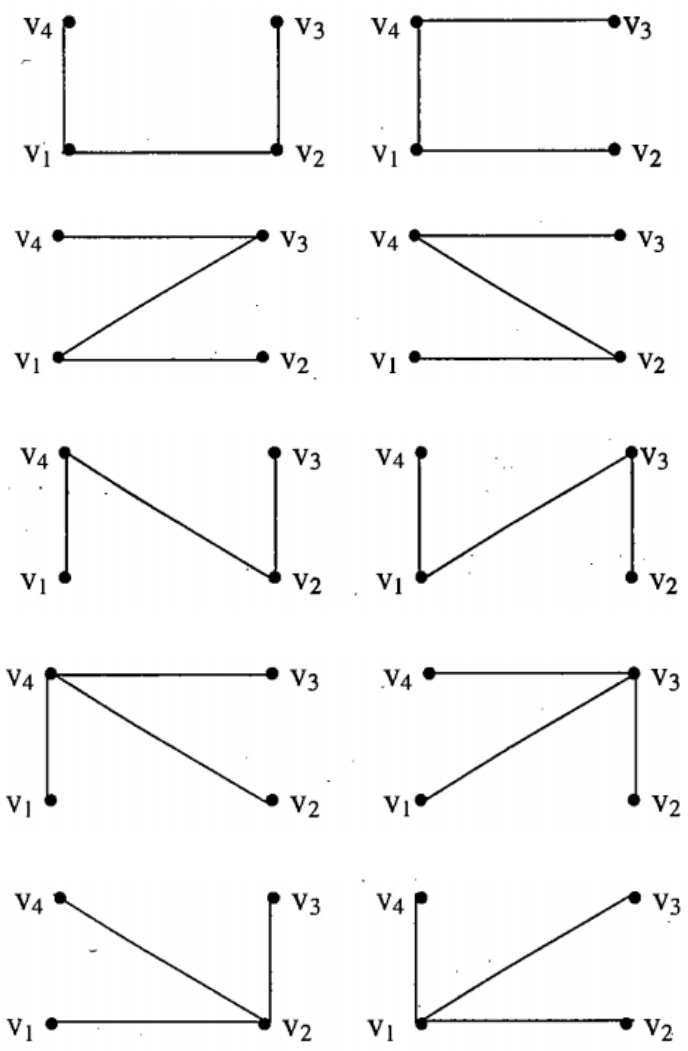
Example: Find all the spanning trees of K_4.

Solution: According to Cayley's formula, K_4 has $4^{4-2} = 4^2 = 16$ different spanning trees.



Here n = 4, so the number of edges in any tree should be n –1 = 4 – 1 = 3. But here number of edges is equal to 6. So to get a tree, we have to remove three edges of K_4. The 16 spanning trees so obtained are shown below:





8.12 Minimal Spanning Tree

Definition: Let G be a weighted graph. A spanning tree of G with minimum weight is called **minimal spanning tree of G.**

We discuss two algorithms to find a minimal spanning tree for a weighted graph G.

8.12.1 Prim Algorithm

Prim algorithm builds a minimal spanning tree T by expanding outward in connected links from some vertex. In this algorithm, one edge and one vertex are added at each step. The edge added is the one of least weight that connects the vertices already in T with those not in T.

Algorithm

Input: A connected weighted graph G with n vertices

Output: The set of edges E in a minimal spanning tree.

Steps of Algorithm

1. Choose a vertex v_1 of G. Let V = $\{v_1\}$ and E = { }.

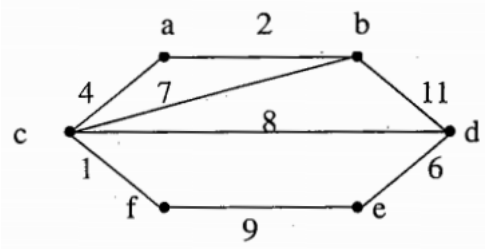
2. Choose a nearest neighbour v_i of V that is adjacent to v_j, where $v_j \in V$ and for which the edge (v_i, v_j) does not form a cycle with members of E. Add v_i to V and add (v_i, v_j) to E.

3. Repeat step 2 till number of edges in T is n – 1. Then V contains all n vertices of G and E contains the edges of a minimal spanning tree for G.

Definition: A **greedy algorithm** is an algorithm that optimizes the choice at each iteration without regard to previous choices.

For example, **Prim algorithm is a greedy algorithm.**

Example: Find a minimal spanning tree for the graph shown below:



Solution: We shall use Prim algorithm to find the required minimal spanning tree. We see that number of vertices in this connected weighted graph is 6. Therefore the tree will have 5 edges.

We start with any vertex, say c. The nearest neighbour of c is f and (c, f) does not form a cycle. Therefore (c, f) is the first edge selected.

Now we consider the set of vertices V = {c, f}. The vertex a is nearest neighbour to V = {c, f} and the edge (c, a) does not form a cycle with the member of set of edges selected so far. Thus

E = {(c, f), (c, a)} and V = {c, f, a}.

The vertex b is now nearest neighbour to V = {c, f, a} and the edge (a, b) do not form a cycle with the member of E = {(c, f), (c, a)}. Thus

E = {(c, f), (c, a), (a, b)} and V = {c, f, a, b}

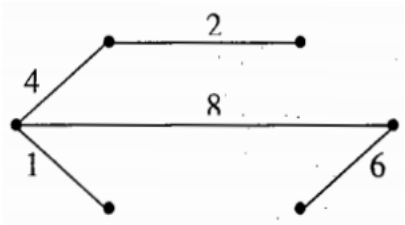
Now the edge (b, c) cannot be selected because it forms a cycle with the members of E. We see that d is the nearest point to V = {c, f, a, b} and (c, d) is the edge which does not form a cycle with members of E = {(c, f), (c, a), (a, b)}. Thus we get

E = {(c, f), (c, a), (a, b), (c, d)}, V = {c, f, a, b, d}

The nearest vertex to V is now e and (d, e) in the corresponding edge. Thus

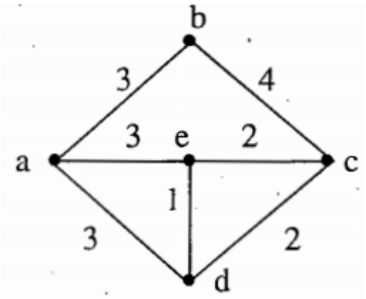
E = {(c, f), (c, a), (a, b), (c, d), (d, e)}, V = {c, f, a, b, d, e}

Since number of edges in the Prim Tree is 5, the process is complete. The minimal spanning tree is shown below:



The length of the minimal spanning tree is 1 + 4 + 2 + 8 + 6 = 21

Example: Using Prim algorithm, find the minimal spanning tree of the following graph:



Solution: Pick up the vertex a. Then

$E = \{\ \}$ and $V = \{a\}$.

The nearest neighbour of V is b, e or d and the corresponding edges are (a, b) or (a, d) or (a, e). We choose arbitrarily (a, b) and have

$E = \{(a, b)\}$, $V = \{a, b\}$

Now d is the nearest neighbour of $V = \{a, b\}$ and the corresponding edge (a, d) does not form cycle with (a, b). Thus we get

$E = \{(a, b), (a, d)\}$, $V = \{a, b, d\}$.

Now e is the nearest neighbour of {a, b, d} and (d, e) does not form cycle with {(a, b), (c, d)}. Hence

$E = \{(a, b), (a, d), (d, e)\}$, $V = \{a, b, d, e\}$

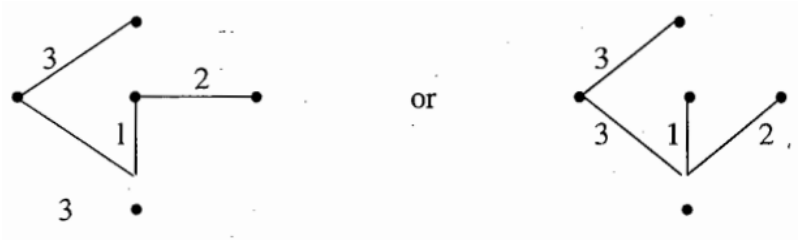
Now c is the nearest neighbour of $V = \{a, b, d, e\}$ and the corresponding edges are (e, c), (d, c). Thus we have, choosing (e, c),

$E = \{(a, b), (a, d), (d, e), (e, c)\}$, $V = \{a, b, d, e, c\}$

Total weight = 3 + 3 + 1 + 2 = 9

(If we choose (d, c), then total weight is 3 + 3 + 1 + 2 = 9)

The minimal spanning tree is



8.12.2 Kruskal's Algorithm

In Kruskal's algorithm, the edges of a weighted graph are examined one by one in order of increasing weight. At each stage an edge with least weight out of edge-set remaining at that

stage is added provided this additional edge does not create a circuit with the members of existing edge set at that stage. After n – 1 edges have been added, these edges together with the n vertices of the connected weighted graph form a minimal tree.

Algorithm

Input: A connected weighted graph G with n vertices and the set E = $\{e_1, e_2, \ldots\ldots, e_k\}$ of weighted edges of G.

Output: The set of edges in a minimal spanning tree T for G.

Steps of Algorithm

Step 1. Initialize T to have all vertices of G and no edges.

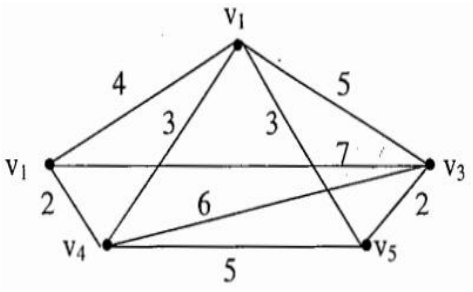
Step 2. Choose an edge e_1 in E of least weight. Let

$E^* = \{e_1\}, \quad E = E - \{e_1\}$

Step 3. Select an edge e_i in E of least weight that does not form circuit with members of E*. Replace E* by $E^* \cup \{e_i\}$ and E with $E - \{e_i\}$.

Step 4. Repeat step 3 until number of edges in E* is equal to n –1.

Example: Use Kruskal's algorithm to determine a minimal spanning tree for the connected weighted graph G shown below.



Solution: The given weighted graph has five vertices. The minimal spanning tree would have therefore 4 edges.

Let

$E = \{(v_1, v_2), (v_1, v_4), (v_1, v_5), (v_2, v_3), (v_1, v_3), (v_2, v_4), (v_4, v_5), (v_5, v_3), (v_3, v_4)\}$

The edges (v_2, v_4) and (v_3, v_5) have minimum weight. We choose arbitrarily one of these, say, (v_2, v_4). Thus

$$E^* = \{(v_2, v_4)\},$$

$$E = E - \{(v_2, v_4)\}.$$

The edge (v_3, v_5) has minimum weight, so we pick it up. We have thus

$$E^* = \{(v_2, v_4), (v_3, v_5)\},$$

$$E = E - \{(v_5, v_4), (v_3, v_5)\}$$

The edges (v_1, v_4) and (v_1, v_5) have minimum weight in the remaining edge set. We pick (v_1, v_4) say, as it does not form a cycle with E*. Thus

$$E^* = \{(v_2, v_4), (v_3, v_5), (v_1, v_4)\},$$

$$E = E - \{(v_2, v_4), (v_3, v_5), (v_1, v_4)\}$$

Now the edge (v_1, v_5) has minimum weight in $E \setminus \{(v_1, v_4), (v_3, v_5), (v_1, v_4)\}$and it does not form a cycle with E*. So, we have

$$E^* = \{(v_2, v_4), (v_3, v_5), (v_1, v_4), (v_1, v_5)\}$$

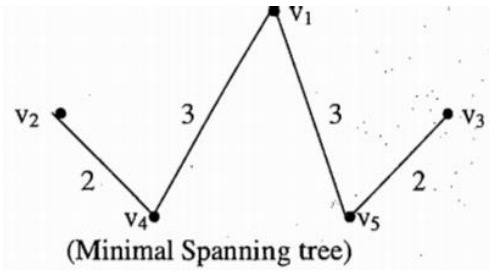
and

$$E = E - \{(v_2, v_4), (v_3, v_5), (v_1, v_4), (v_1, v_5)\}$$

Thus all the four edges have been selected. The minimal tree has the edges

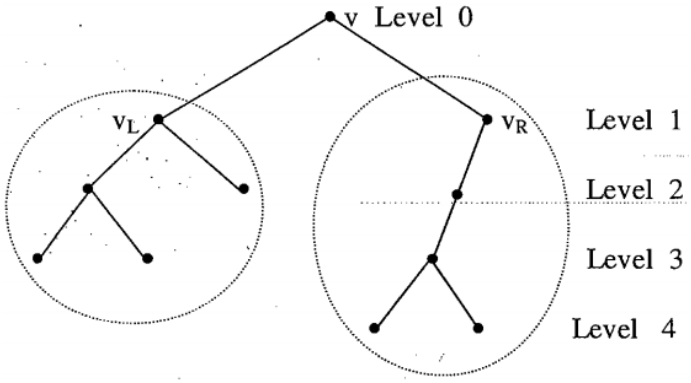
$$(v_2, v_4), (v_3, v_5), (v_1, v_4), (v_1, v_5)$$

and is shown below :



8.13 Tree Searching

Let T be a binary tree of height h ≥ 1 and root v. Since h ≥ 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.**

8.14 Different Methods of Searching a Tree

We discuss three different methods of searching a tree, namely preorder search method, postorder search method and inorder search method.

8.14.1 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

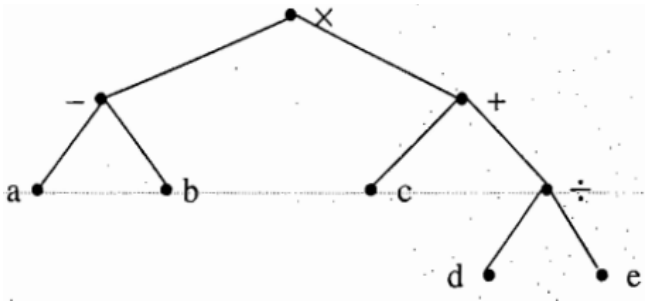
Step 3. Search the right subtree if it exists.

Example 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, × 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

$$\times - a\ b + c \div d\ e$$

This form of the expression is called **prefix form** or **polish form** of the expression

$(a - b) \times (c + (d \div e))$

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 in Polish 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.

Example: Find parenthesized form of the polish expression

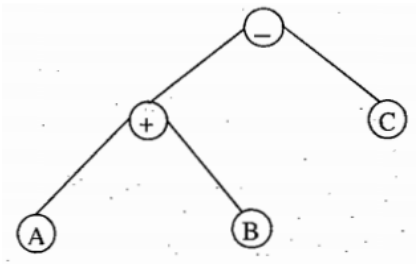
– + A B C

Solution: The parenthesized form of the given polish expression is derived a follows:

– (A + B) C

(A + B) – C

The corresponding binary tree is



8.14.2 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.

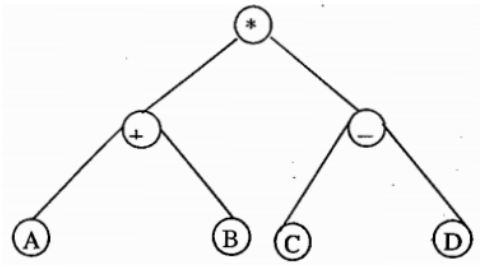
End of algorithm.

Example: 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.

Example: Find the parenthesized form of the postfix form

A B C * * C D E + / –

Solution: We have

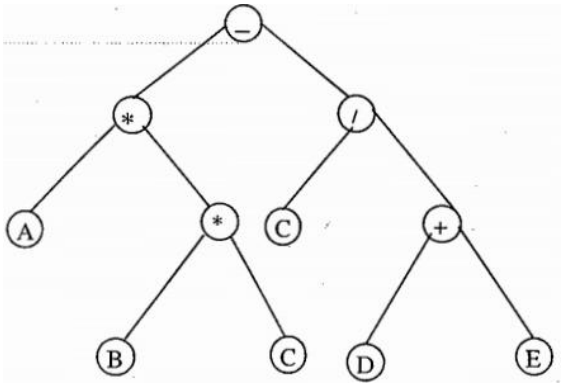
1. A B C * * C D E + / –

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



Example: 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

8.15 Check Your Progress

1. A tree has five vertices of degree 2, three vertices of degree 3 and four vertices of degree 4. How many vertices of degree 1 does it have ?

2. A tree has 2n vertices of degree 1, 3n vertices of degree 2 and n vertices of degree

3. Determine the number of vertices and edges in the tree.

3. If T = (V, E) be a rooted tree with v0 as its root then T is a acyclic.

4. If in a graph G there is one and only one path between every pair of vertices, G.

5. A simple graph G has a spanning tree if and only if G is connected.

8.16 Summary

In this chapter, we have discussed about definition and properties of trees, directed trees, rooted trees, binary trees, spanning tree of a graph, minimal spanning tree. Further, we have studied some algorithms, namely, Prim algorithm and Kruskal's algorithm for finding minimal spanning tree for a weighted graph. We have also discussed different methods of searching a tree. Examples have been given to illustrate these topics.

8.17 Keywords

Tree, directed trees, rooted trees, binary trees, minimal spanning tree, tree searching

8.18 Self-Assessment Test

1. Every non trivial tree contains atleast two end vertices.

2. If G is a tree and if any two non adjacent vertices of G are joined by an edge e, then G + e has exactly one cycle.

3. If u and v are distinct vertices of a tree T contains exactly one u – v path.

4. Construct two non-isomorphic trees having exactly 4 pendant vertices on 6 Vertices.

5. Construct three distinct trees with exactly
(i) one central vertex

(ii) two central vertices.

8.19 Answers to check your progress

1. Let x be the number of nodes of degree one.
Thus, total number of vertices

$$= 5 + 3 + 4 + x = 12 + x.$$

The total degree of the tree = $5 \times 2 + 3 \times 3 + 4 \times 4 + x = 35 + x$
Therefore number of edges in the three is half of the total degree of the tree.
If G = (V, E) be the tree, then, we have

$$| V | = 12 + x \text{ and } | E | = (35+x)/2$$

In any tree, $| E | = | V | - 1$.
Therefore, we have

$$(35+x)/2 = 12 + x - 1$$
$$\Rightarrow 35 + x = 24 + 2x - 2$$
$$\Rightarrow x = 13$$

Thus, there are 13 nodes of degree one in the tree.

2. It is given that total number of vertices in the tree is $2n + 3n + n = 6n$.
The total degree of the tree is $2n \times 1 + 3n \times 2 + n \times 3 = 11n$.
The number of edges in the tree will be half of $11n$.

If G = (V, E) be the tree then, we have

$$| V | = 6n \text{ and } | E | = (11n)/2$$

In any tree, $| E | = | V | - 1$.

Therefore, we have

$$(11n)/2 = 6n - 1$$
$$\Rightarrow 11n = 12n - 2$$
$$\Rightarrow n = 2$$

Thus, there are $6 \times 2 = 12$ nodes and 11 edges in the tree.

3. Let there is a cycle π in T that begins and end at a node v.
Since the in degree of root is zero, $v \neq v0$.
Also by the definition of tree, there must be a path from $v0$ to v, let it be p.
Then πp is also a path, distinct from p, from $v0$ to v.
This contradicts the definition of a tree that there is unique path from root to every other node. Hence T cannot have a cycle in it. *i.e.*, a tree is always acyclic.

4. Since there exists a path between every pair of vertices then G is connected.
A cycle in a graph (with two or more vertices) implies that there is atleast one pair of vertices u, v such that there are two distinct paths between u and v.
Since G has one and only one path between every pair of vertices, G can have no cycle. Therefore, G is a tree.

5. First, suppose that a simple graph G has a spanning tree T. T contains every vertex of G. Let a and b be vertices of G. Since a and b are also vertices of T and T is a tree, there is a path P between a and b. Since T is subgraph, P also serves as path between a and b in G. Hence G is connected.
Conversely, suppose that G is connected.
If G is not a tree, it must contain a simple circuit. Remove an edge from one of these simple circuits. The resulting subgraph has one fewer edge but still contains all the vertices of G and is connected. If this subgraph is not a tree, it has a simple circuit, so as before, remove an edge that is in a simple circuit.
Repeat this process until no simple circuit remain.
This is possible because there are only a finite number of edges in the graph, the process terminates when no simple circuits remain.
Thus we eventually produce an acyclic subgraph T which is a tree. The tree is a spanning tree since it contains every vertex of G.

8.20 References/ Suggestive Readings

1 J.P. Tremblay & R. Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co., 1997.
2 Seymour Lepschutz, Finite Mathematics (International edition 1983), McGraw-Hili Book Company, New York.
3 C.L. Liu, Elements of Discrete Mathematics, McGraw-

Hili Book Co.

4 N.Deo, Graph Theory with Applications to Engineering and Computer Sciences, Prentice Hall of India.

NOTES

NOTES

NOTES

NOTES