RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING AND TECHNOLOGY

MATHEMATICAL FOUNDATION OF COMPUTER SCIENCE NOTES

(A0504193)

II B.Tech – I SEM (R19)

Department of Computer Science and Engineering

UNIT-I

Mathematical Logic

Syllabus

Mathematical Logic: Statements and notations, Connectives, Well-formed formulas, Truth Tables, tautology, converse, inverse and contrapositive ,equivalence, implication, Normal forms.

Introduction to Discrete Mathematics

- Discrete maths are the study of discrete structures which mathematical models dealing with discrete objects and relationships between them.
- Example of discrete objects are like Sets,Permutations,grpahs and etc.

Why it is important for computer science

- In the world of computers, all the information is stored in bits, units of information that can take the value of either 0 or 1. It's not like in nature, where something can take all the values in between 0 and 1 as well. Instead, everything is binary.
- Since the bits are the building blocks of everything that happens in computer software, everything becomes discrete. For instance, the hard drive on the laptop I'm using right now can store 1 845 074 329 600 bits of information.
- The study of algorithms is also firmly in the discrete world. An algorithm is a step-by-step list of instructions to the computer and it's what makes computer programs possible. When determining how much time an algorithm needs to run, you count the number of operations it needs to perform. Notice the word count. Again, discrete mathematics.

- In continuous mathematics (the opposite of discrete), the calculation would go like this:
- $\int_0^5 x\,dx=[1/2*x2]^5{}_0=5^2/2-0=12.5$
- In discrete mathematics, the equivalent calculation would go like this:$\sum_0^4 Xi$ =0+1+2+3+4=10

Applications of discrete mathematics with computer applications

- 1)Computer networks
- 2)Programming languages
- 3)Finite state automata or compilers
- 4)Databases

Symbolic logic is used in framing algorithms and their verification and in automatic theorem proving. Set theory, Graph Theory, trees etc are used in storage and retrieval of information (data structures),Algorithms and their complexity studies also uses tools from discrete mathematics. Formal Languages, Automata theory, Turing machines etc are themselves part of discrete mathematics and so is Recursive Function Theory.Undecidability of many problems are established using Turing machines which is the Mathematical model for studying theoretical limitations of Computation.Lattices and Boolean Algebra are used in Computer Science as well as in communications and networking.

Mathematical Logic

Logic

It is the study of the principles and methods that distinguishes between valid and invalid argument.

1.Proposition Logic

Proposition or Statement

A proposition or a statement can defined has a declarative sentence to which we can assign one and only one of the truth values either true (or) false but not a both is called a proposition.

The true or false of a proposition is called truth value of a proposition

These two values true and false are denoted by the symbols *T* and *F* respectively. Sometimes these are also denoted by the symbols 1 and 0 respect.

Proposition	truth value
Ex: 1) India capital is new Delhi	True
2) 2*3=5	false
3) 5 is a prime number	true

These are propositions (or statements) because they are either true or false.

Next consider the following sentences:

4) How beautiful are you?
5) Wish you a happy new year
6) $x + y = z$
7) Take one book.
These are not propositions as they are not declarative in nature, that is, they do not declare a definite truth value *T* or *F*.

Types of Propositions

1)Atomic proposition

2)Compound Proposition

1)Atomic proposition

- A Proposition which cannot be divided further is called an atomic proposition .
- **Examples:**
- 1)India capatial is New Delhi
- 2)2*3=5

2) Compound Proposition

- Two or more atomic propositions can be combined to form a compound proposition with help of Connectives. Compound Proposition also called as Propositional function.

- **Examples of Compound statements**
- "3+2=5" and "delhi is a capatial of india".
- "the grass is green" or "it is hot today".
- Discrete mathematics is not difficult to me.
- Here and, or ,not are called connectives.

Notations

- Statements are symbollically represented as A,B,C,..P,Q,R,S… Those are called propostional variables or notations.
- **Examples**:
- P="Delhi is the capital of india".
- Q="17 is divisible by 3.

Logical Connectives

- The words or phrases or symbols which are used to make a compound proposition by two or more atomic propositions are called **logical connectives** or **simply connectives.**
- There are five basic connectives called negation, conjunction, disjunction, conditional and biconditional.

Connectivity	**Symbol**	Word
Negation	~	Not
Disjunction	∨	OR
Conjunction	∧	AND
Conditional	→	IF AND THEN
Biconditional	↔	If and only if

Negation (~) or(¬)

The **negation** of a statement is generally formed by writing the word ‘not‘ at a proper place in the statement (proposition) or by prefixing the statement with the phrase (~).It is not the case that‘. If *p* denotes a statement then the negation of *p* is written as *p* and read as not *p*‘. If the truth value of *p* is *T* then the truth value of *p* is *F*. Also if the truth value of *p* is *F* then the truth value of *p* is *T*.

Ex:~P

Truth table for Negation

P	~P
F	T

Disjunction (OR)

- If P and Q are any two propositions then P or Q symbollicaly written as P V Q .
- P V Q is a proposition whose truth values is false only when both P and Q are false other wise True.

Truthtable for Disjunction

P	Q	P V Q
F	F	F
F	T	T
T	F	T
T	T	T

P: I shall go to the game.

Q : I shall watch the game on television.

P V Q: I shall go to the game OR I shall watch the game on television.

Conjunction (AND(^))

- IF P and Q are any two propositions then P and Q Symbollically written as P ∧ Q.
- P ∧ Q is a proposition whose truth value is true only when both are P and Q are true.
- Whole truth value is false when either P or Q are false and both are false.

Truth table for Conjunction

P	Q	P ∧ Q
F	F	F
F	T	F
T	F	F
T	T	T

P : It is raining today.

Q: There are 10 chairs in the room.

P ∧ Q: It is raining today AND There are 10 chairs in the room.

Conditional (or) Implication(→)

If P and Q are any two statements (or propositions) then the statement P → Q which is read as, If P , then Q‘ is called a **conditional statement** (or **proposition**) or **implication** and the connective is the **conditional connective.**

Truth table for conditional

P	Q	P→ Q
F	F	T
F	T	T
T	F	F
T	T	T

In this conditional statement, P is called the **hypothesis** or **premise** or **antecedent** and Q is called the **consequence** or **conclusion.**

Biconditional (If and only if)

- If P and Q are any two propostions then P if and only if Q Written as P ↔Q.
- It's truth value is true only when both P & Q have same truth values.

P	*q*	p↔q
T	*T*	*T*
T	*F*	*F*
F	*T*	*F*
F	*F*	*T*

TAUTOLOGY AND CONTRADICTION

Tautology: A proposition is said to be a tautology if its truth value is T for any assignment of truth values to its components.

Example: :The proposition P V ~P is a tautology.

Contradiction

A proposition is said to be a contradiction if its truth value is F for any assignment of truth values to its components. Example: The proposition P ∧ ¬Pis a contradiction.

Contingency: A statement formula which is neither a tautology nor a contradiction is known as a **contingency.**

Example: P->Q

Tautology: A statement formula which is true regardless of the truth values of the statements which replace the variables in it is called a **universally valid formula** or a **logical truth** or a **tautology.**

How to prove given compound proposition is tautology

1)By Constructing truth table

2)By using substitution method

1)Constructing truth table

Show that following function is tautology

a)(PVQ) V ~P

Solution

P	Q	~P	PVQ	(PVQ)V~P
F	F	T	F	T
F	T	T	T	T
T	F	F	T	T
T	T	F	T	T

In the above table **(PVQ) V ~P** is givig all truth values are true so it is a tautology.

b)Show that given propostion function is a tautology ((P → Q) ∧ (Q→R)) →(P→ R)

P	Q	R	P → Q	Q→R	((P → Q) ∧ (Q→R)	P→ R	((P → Q) ∧ (Q→R)) →(P→ R)
F	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T
F	T	F	T	F	F	T	T
T	F	F	F	T	F	F	T
F	T	T	T	T	T	T	T
T	F	T	F	T	F	T	T
T	T	F	T	F	F	F	T
T	T	T	T	T	T	T	T

Implication:

- If P and Q are any two propositions then P->Q or If P then Q
- P->Q is proposition whose truth value is false only when P is true and Q is false.
- P->Q
- Here P is antecedent and Q is Consequent.

P	Q	P→ Q
F	F	T
F	T	T
T	F	T
T	T	T

- When ever P is false P->Q is true
- i.e false antecedent P implies any proposition Q
- When ever Q is true P->Q is also true
- i.e a true consequent Q implied by any propositional 'P'.
  from the implication statement we can write another three statements which are converse, inverse and contrapositive

Converse, Inverse and Contrapositive

If $P \rightarrow Q$ is a conditional statement, then

(1). $Q \rightarrow P$ is called its *converse*

(2). $\neg P \rightarrow \neg Q$ is called its *inverse*

(3). $\neg Q \rightarrow \neg P$ is called its *contrapositive.*

Example:

- P: **Today is Sunday**
- Q: **It is a holiday**
- **Converse Statement:** If it is a holiday, then today is Sunday**.**
- **Inverse Statement:** If today is not Sunday, then it is not a holiday.
- **Contrapositive Statement**- If it is not a holiday, then today is not Sunday.

Here Implication and Contrapositive are equal.

Converese and inverse are opposite propostions.

Well formed formulas(wff):

Not all strings can represent propositions of the predicate logic. Those which produce a proposition when their symbols are interpreted must follow the rules given below, and they are called wffs(well-formed formulas) of the first order predicate logic.

Rules for constructing Wffs

A predicate name followed by a list of variables such as P(x, y), where P ispredicate name, and x and y are variables, is called an atomic formula.

A well formed formula of predicate calculus is obtained by using the following rules.

1. An atomic formula is a wff.
2. If A is a wff, then 7A is also a wff.
3. If A and B are wffs, then (A V B), (A ^ B), (A → B) and (A D B).
4. If A is a wff and x is a any variable, then (x)A and ($x)A are wffs.
5. Only those formulas obtained by using (1) to (4) are wffs.

Since we will be concerned with only wffs, we shall use the term formulas for wff. We shall follow the same conventions regarding the use of parentheses as was done in the case of statement formulas.

Wffs are constructed using the following rules:

1. *True* and *False* are wffs.
2. Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.
3. Each atomic formula (i.e. a specific predicate with variables) is a wff.
4. If *A*, *B*, and *C* are wffs, then so are $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$.
5. If *x* is a variable (representing objects of the universe of discourse), and *A* is a wff, then so are $\forall x\, A$ and $\exists x\, A$.

For example, "The capital of Virginia is Richmond." is a specific proposition. Hence it is a wff by Rule

2.

Let B be a predicate name representing "being blue" and let x be a variable. Then B(x) is an atomic formula meaning "x is blue". Thus it is a wff by Rule 3. above. By applying Rule 5. to B(x), ∀xB(x) is a wff and so is ∃xB(x). Then by applying Rule 4. to them ∀x B(x) ∧∃x B(x) is seen to be a wff. Similarly if R is a predicate name representing "being round". Then R(x) is an atomic formula. Hence it is a wff. By applying Rule 4 to B(x) and R(x), a wff B(x) ∧R(x) is obtained. In this manner, larger and more complex wffs can be constructed following the rules given above. Note, however, that strings that can not be constructed by using those rules are not wffs. For example, ∀xB(x)R(x), and B(∃x) are NOT wffs, NOR are B(R(x)), and B(∃x R(x)) .

More examples: To express the fact that Tom is taller than John, we can use the atomic formula ***taller*(Tom, John)**, which is a wff. This wff can also be part of some compound statement such as ***taller*(Tom, John)** ∧¬***taller*(John, Tom)**, which is also a wff.
If *x* is a variable representing people in the world, then ***taller(x,*Tom),** ∀***x taller(x,*Tom),** ∃***x taller(x,*Tom),** ∃***x*** ∀***y taller(x,y)*** are all wffs among others. However, *taller(*∃ *x,*John) and *taller*(Tom ∧Mary, Jim), for example, are **NOT** wffs.

Logical Equivalence

> Two formulas *A* and *B* are said to equivalent to each other if and only if *A*↔ *B* is a tautology.If *A*↔*B* is a tautology, we write *A* ⇔ *B* which is read as *A* is equivalent to *B.*

Note : 1. ⇔ is only symbol, but not connective.

A ↔ *B* is a tautology if and only if truth tables of *A* and *B* are the same. Equivalence relation is symmetric and transitive.

(or)

Let P and Q are two propositional functions P is Equivalent to Q.

Symbolically written as P⇔ Q or P ≡ Q if P and Q have same truth table.

Ex:P->Q ⇔ ~P V Q.

Method I. Truth Table Method: One method to determine whether any two statement formulas are equivalent is to construct their truth tables.

Example: Prove $(P \rightarrow Q) \Leftrightarrow (\neg P \vee Q)$.

P	Q	$P \rightarrow Q$	$\neg P$	$\neg P \vee Q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

In the above table both $P \rightarrow Q$ *and* $\neg P \vee Q$ *are have same truth values.*

So that $(P \rightarrow Q) \Leftrightarrow (\neg P \vee Q)$.

Equivalence Formulas:

1. Idempotent laws:

   (a) $P \vee P \Leftrightarrow P$ (b) $P \wedge P \Leftrightarrow P$

2. Associative laws:

   (a) $(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$ (b) $(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$

3. Commutative laws:

   (a) $P \vee Q \Leftrightarrow Q \vee P$ (b) $P \wedge Q \Leftrightarrow Q \wedge P$

4. Distributive laws:

   $P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$ $P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$

   Identity laws (or)

5. Domination Law

   (a) (i) $P \vee F \Leftrightarrow P$ (ii) $P \vee T \Leftrightarrow T$

(b) (i) $P \wedge T \Leftrightarrow P$ (ii) $P \wedge F \Leftrightarrow F$

6. Component laws:

(a) (i) $P \vee \neg P \Leftrightarrow T$ (ii) $P \wedge \neg P \Leftrightarrow F$.

(b) (i) $\neg\neg P \Leftrightarrow P$ (ii) $\neg T \Leftrightarrow F$, $\neg F \Leftrightarrow T$

7. Absorption laws:

(a) $P \vee (P \wedge Q) \Leftrightarrow P$ (b) $P \wedge (P \vee Q) \Leftrightarrow P$

8.Demorgan's Laws

(a) $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$ (b) $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

TABLE 7 Logical Equivalences Involving Conditional Statements.

$$p \to q \equiv \neg p \vee q$$

$$p \to q \equiv \neg q \to \neg p$$

$$p \vee q \equiv \neg p \to q$$

$$p \wedge q \equiv \neg(p \to \neg q)$$

$$\neg(p \to q) \equiv p \wedge \neg q$$

$$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$$

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

$$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$$

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

Tautological Implications.

A statement formula A is said to *tautologically imply* a statement B if and only if $A \rightarrow B$ is a tautology.In this case we write $A \Rightarrow B$, which is read as 'A implies B'.

Note: $\Rightarrow$ is not a connective, $A \Rightarrow B$ is not a statement formula.

$A \Rightarrow B$ states that $A \rightarrow B$ is tautology.

Clearly $A \Rightarrow B$ guarantees that B has a truth value T whenever A has the truth value T .One can determine whether $A \Rightarrow B$ by constructing the truth tables of A and B in the same manner as was done in the determination of $A \Leftrightarrow B$.

Example: Prove that $(P \rightarrow Q) \Rightarrow (\neg Q \rightarrow \neg P)$.

P	Q	$\neg P$	$\neg Q$	$P \rightarrow Q$	$\neg Q \rightarrow \neg P$	$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$
T	T	F	F	T	T	T
T	F	F	T	F	F	T
F	T	T	F	T	T	T
F	F	T	T	T	T	T

Since all the entries in the last column are true, $(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$ is a tautology.

Hence $(P \rightarrow Q)$ is Tautological Implications to $(\neg Q \rightarrow \neg P)$.

So that $(P \rightarrow Q) \Rightarrow (\neg Q \rightarrow \neg P)$.

In order to show any of the given implications, it is sufficient to show that an assignment of the truth value T to the antecedent of the corresponding condi-tional leads to the truth value T for the

consequent. This procedure guarantees that the conditional becomes tautology, thereby proving the implication.

Example: Prove that $\neg Q \wedge (P \rightarrow Q) \Rightarrow \neg P$.

Solution: Assume that the antecedent $\neg Q \wedge (P \rightarrow Q)$ has the truth value T , then both $\neg Q$ and $P \rightarrow Q$ have the truth value T , which means that Q has the truth value F , $P \rightarrow Q$ has the truth value T .

Hence P must have the truth value F .Therefore the consequent $\neg P$ must have the truth value T.

$\neg Q \wedge (P \rightarrow Q) \Rightarrow \neg P$.

Another method to show $A \Rightarrow B$ is to assume that the consequent B has the truth value F and then show that this assumption leads to A having the truth value F . Then $A \rightarrow B$ must have the truth value T .

Example: Show that $\neg(P \rightarrow Q) \Rightarrow P$.

Solution: Assume that P has the truth value F . When P has F , $P \rightarrow Q$ has T , then $\neg(P \rightarrow Q)$ has F.Hence $\neg(P \rightarrow Q) \rightarrow P$ has T .

So that $\neg(P \rightarrow Q) \Rightarrow P$.

Normal Forms

If a given statement formula $A(p_1, p_2, ...p_n)$ involves n atomic variables, we have 2^n possible combinations of truth values of statements replacing the variables.

The formula A is a tautology if A has the truth value T for all possible assignments of the truth values to the variables $p_1, p_2, ...p_n$ and A is called a contradiction if A has the truth value F for all possible assignments of the truth values of the n variables. A is said to be *satisable* if A has the truth value T for atleast one combination of truth values assigned to $p_1, p_2, ...p_n$.

The problem of determining whether a given statement formula is a Tautology, or a Contradiction is called a decision problem.

The construction of truth table involves a finite number of steps, but the construction may not be practical. We therefore reduce the given statement formula to normal form and find whether a given statement formula is a Tautology or Contradiction or at least satisfiable.It will be convenient to use the word product in place of conjunction and sum in place of disjunction .

A product of the variables and their negations in a formula is called an *elementary Product.*
Similarly, a sum of the variables and their negations in a formula is called an *elementary sum.*
Let P and Q be any atomic variables Then $P,\sim P\wedge Q,\ \neg Q\wedge P\wedge\sim P,Q\wedge\sim P$ are some example of elementary products.
On the other hand $P,\sim P\vee Q,\ \neg Q\vee P\vee\sim P,Q\vee\sim P.$ are some examples of elementary sums.
Types of Normal forms
1) Disjunctive Normal forms
2) Conjunctive Normal forms.

Disjunctive Normal Form (DNF)

A formula which is equivalent to a given formula and which consists of a sum of elementary products is called a *disjunctive normal form* of the given formula.

Example: Obtain disjunctive normal forms of

(a) $P \wedge (P \rightarrow Q)$

$P\wedge(P\rightarrow Q)\Leftrightarrow P\wedge(\neg P\vee Q)$ (apply distributive law $P\wedge(Q\vee R)\Leftrightarrow(P\wedge Q)\vee(P\wedge R)$

$\Leftrightarrow (P\wedge\neg P)\vee(P\wedge Q)$

b) $\neg(P\vee Q)\leftrightarrow(P\wedge Q)$

$\neg(P\vee Q)\leftrightarrow(P\wedge Q)\Leftrightarrow(\neg(P\vee Q)\wedge(P\wedge Q))\vee((P\vee Q)\wedge\neg(P\wedge Q))$ [using $R\leftrightarrow S\Leftrightarrow(R\wedge S)\vee(\neg R\wedge\neg S)$]

$\Leftrightarrow((\neg P\wedge\neg Q)\wedge(P\wedge Q))\vee((P\vee Q)\wedge(\neg P\vee\neg Q)).$

$\Leftrightarrow(\neg P\wedge\neg Q\wedge P\wedge Q)\vee((P\vee Q)\wedge\neg P)\vee((P\vee Q)\wedge\neg Q).$

$\Leftrightarrow(\neg P\wedge\neg Q\wedge P\wedge Q)\vee(P\wedge\neg P)\vee(Q\wedge\neg P)\vee(P\wedge\neg Q)\vee(Q\wedge\neg Q).$

Which is the required disjunctive normal form.Note: The DNF of a given formula is not unique.

Conjunctive Normal Form (CNF)

A formula which is equivalent to a given formula and which consists of a product of elementary sums is called a *conjunctive normal form* of the given formula.

The method for obtaining conjunctive normal form of a given formula is similar to the one given for disjunctive normal form. Again, the conjunctive normal form is not unique.

(a) $P \wedge (P \rightarrow Q)$ obtain the conjuctive normal form

$P \wedge (P \rightarrow Q) \Leftrightarrow P \wedge (\neg P \vee Q)$

(b). $\neg(P \vee Q) \leftrightarrow (P \wedge Q)$

$$\Leftrightarrow (\neg(P \vee Q) \rightarrow (P \wedge Q)) \wedge ((P \wedge Q) \rightarrow \neg(P \vee Q))$$
$$\Leftrightarrow ((P \vee Q) \vee (P \wedge Q)) \wedge (\neg(P \wedge Q) \vee \neg(P \vee Q))$$
$$\Leftrightarrow [(P \vee Q \vee P) \wedge (P \vee Q \vee Q)] \wedge [(\neg P \vee \neg Q) \vee (\neg P \wedge \neg Q)]$$
$$\Leftrightarrow (P \vee Q \vee P) \wedge (P \vee Q \vee Q) \wedge (\neg P \vee \neg Q \vee \neg P) \wedge (\neg P \vee \neg Q \vee \neg Q)$$

Principal Disjunctive Normal Form

In this section, we will discuss the concept of principal disjunctive normal form (PDNF).

Minterm: For a given number of variables, the minterm consists of conjunctions in which each statement variable or its negation, but not both, appears only once.

Let P and Q be the two statement variables. Then there are 2^2 minterms given by $P \wedge Q$, $P \wedge \neg Q$,

$\neg P \wedge Q$, and $\neg P \wedge \neg Q$

Minterms for three variables P , Q and R are $P \wedge Q \wedge R$, $P \wedge Q \wedge \neg R$, $P \wedge \neg Q \wedge R$,$P \wedge \neg Q \wedge \neg R$,

$\neg P \wedge Q \wedge R$, $\neg P \wedge Q \wedge \neg R$, $\neg P \wedge \neg Q \wedge R$ and $\neg P \wedge \neg Q \wedge \neg R$.

From the truth tables of these minterms of *P* and *Q*, it is clear that.

P	*Q*	$P \wedge Q$	$P \wedge \neg Q$	$\neg P \wedge Q$	$\neg P \wedge \neg Q$
T	T	T	F	F	F
T	F	F	T	F	F
F	T	F	F	T	F
F	F	F	F	F	T

(i). no two minterms are equivalent

(ii). Each minterm has the truth value *T* for exactly one combination of the truth values of the variables *P* and *Q*.

PDNF

Definition: For a given formula, an equivalent formula consisting of disjunctions of minterms only is called the Principal disjunctive normal form of the given formula. The principle disjunctive normal formula is also called the sum-of-products canonical form.

Methods to obtain the PDNF of a given formula.

(a). By Truth table:

(b). without constructing the truth table

(a). By Truth table:

(i). Construct a truth table of the given formula.

(ii). For every truth value *T* in the truth table of the given formula, select the minterm which also has the value *T* for the same combination of the truth values of *P* and *Q*.

(iii). The disjunction of these minterms will then be equivalent to the given formula

Example: Obtain the PDNF of $P \rightarrow Q$.

Solution: From the truth table of $P \rightarrow Q$

P	Q	$P \rightarrow Q$	Minterm
T	T	T	$P \wedge Q$
T	F	F	$P \wedge \neg Q$
F	T	T	$\neg P \wedge Q$
F	F	T	$\neg P \wedge \neg Q$

The PDNF of $P \rightarrow Q$ is $(P \wedge Q) \vee (\neg P \wedge Q) \vee (\neg P \wedge \neg Q)$.

Obtain the PDNF for $(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R)$.

Solution:

P	Q	R	Minterm	$P \wedge Q$	$\neg P \wedge R$	$Q \wedge R$	$(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R)$
T	T	T	$P \wedge Q \wedge R$	T	F	T	T
T	T	F	$P \wedge Q \wedge \neg R$	T	F	F	T
T	F	T	$P \wedge \neg Q \wedge R$	F	F	F	F
T	F	F	$P \wedge \neg Q \wedge \neg R$	F	F	F	F
F	T	T	$\neg P \wedge Q \wedge R$	F	T	T	T
F	T	F	$\neg P \wedge Q \wedge \neg R$	F	F	F	F
F	F	T	$\neg P \wedge \neg Q \wedge R$	F	T	F	T
F	F	F	$\neg P \wedge \neg Q \wedge \neg R$	F	F	F	F

The PDNF of $(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R)$ is $(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R)$.

(b). Without constructing the truth table:

In order to obtain the principal disjunctive normal form of a given formula is constructed as follows:

(1). First replace $\rightarrow$, by their equivalent formula containing only $\wedge$, $\vee$ and $\neg$.

(2). Next, negations are applied to the variables by De Morgan's laws followed by the application of distributive laws.

(3). Any elementarily product which is a contradiction is dropped. Minterms are ob-tained in the disjunctions by introducing the missing factors. Identical minterms appearing in the disjunctions are deleted.

Example: Obtain the principal disjunctive normal form of

(*a*) $\neg P \vee Q$
(*b*) $(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R)$.

(*a*) $\neg P \vee Q$

$\neg P \vee Q \Leftrightarrow (\neg P \wedge T) \vee (Q \wedge T)$

$\Leftrightarrow (\neg P \wedge (Q \vee \neg Q)) \vee (Q \wedge (P \vee \neg P)) \quad [\because P \vee \neg P \Leftrightarrow T]$

$\Leftrightarrow (\neg P \wedge Q) \vee (\neg P \wedge \neg Q) \vee (Q \wedge P) \vee (Q \wedge \neg P)$

$\Leftrightarrow (\neg P \wedge Q) \vee (\neg P \wedge \neg Q) \vee (P \wedge Q) \quad [\because P \vee P \Leftrightarrow P]$

(*b*) $(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R)$

$(P \wedge Q) \vee (\neg P \wedge R) \vee (Q \wedge R) \Leftrightarrow (P \wedge Q \wedge T) \vee (\neg P \wedge R \wedge T) \vee (Q \wedge R \wedge T)$

$\Leftrightarrow (P \wedge Q \wedge (R \vee \neg R)) \vee (\neg P \wedge R \wedge (Q \vee \neg Q)) \vee (Q \wedge R \wedge (P \vee \neg P))$

$\Leftrightarrow (P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge R \wedge Q) \vee (\neg P \wedge R \wedge \neg Q) \vee (Q \wedge R \wedge P) \vee (Q \wedge R \wedge \neg P)$

$\Leftrightarrow (P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge R)$

Principal Conjunctive Normal Form

The dual of a minterm is called a Maxterm. For a given number of variables, the *maxterm* consists of disjunctions in which each variable or its negation, but not both, appears only once. Each of the maxterm has the truth value *F* for exactly one combination of the truth values of the variables. Now we define the principal conjunctive normal form.

For a given formula, an equivalent formula consisting of conjunctions of the max-terms only is known as its *principle conjunctive normal form*. This normal form is also called the *product-of-sums canonical form.*The method for obtaining the PCNF for a given formula is similar to the one described previously for PDNF.

Example: Obtain the principal conjunctive normal form of the formula (¬P→R)∧(Q↔P)

Solution:

$(\neg P \rightarrow R) \wedge (Q \leftrightarrow P) \Leftrightarrow [\neg(\neg P) \vee R] \wedge [(Q \rightarrow P) \wedge (P \rightarrow Q)]$

$\Leftrightarrow [(P \vee R) \wedge [(\neg Q \vee P) \wedge (\neg P \vee Q)]$

$\Leftrightarrow (P \vee R \vee F) \wedge [(\neg Q \vee P \vee F) \wedge (\neg P \vee Q \vee F)]$

$\Leftrightarrow [(P \vee R) \vee (Q \wedge \neg Q)] \wedge [\neg Q \vee P) \vee (R \wedge \neg R)] \wedge [(\neg P \vee Q) \vee (R \wedge \neg R)]$

$\Leftrightarrow$

$(P \vee R \vee Q) \wedge (P \vee R \vee \neg Q) \wedge (P \vee \neg Q \vee R) \wedge (P \vee \neg Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee Q \vee \neg R)$

Unit-II

Syllabus:

Predicates: Rules of inference, Consistency, Predicate calculus: Free and bounded variable, Quantifiers: Universal Quantifiers, Existential Quantifiers.

Rules of inference:

The two rules of inference are called rules P and T.

Rule P: A premise may be introduced at any point in the derivation.

Rule T: A formula S may be introduced in a derivation if s is tautologically implied by any one or more of the preceding formulas in the derivation.

Before proceeding the actual process of derivation, some important list of implications and equivalences are given in the following tables.

Implications

I1	P∧Q =>P	}	Simplification
I2	PQ∧ =>Q		
I3	P=>PVQ	}	Addition
I4	Q =>PVQ		
I5	7P => P→ Q		
I6	Q => P→ Q		
I7	7(P→Q) =>P		
I8	7(P → Q) => 7Q		
I9	P, Q => P ∧ Q		
I10	7P, PVQ => Q		(disjunctive syllogism)
I11	P, P→ Q => Q		(modus ponens)
I12	7Q, P → Q => 7P		(modus tollens)
I13	P → Q, Q → R => P → R		(hypothetical syllogism)
I14	P V Q, P → Q, Q → R => R		(dilemma)

Equivalences

E1 77P <=>P

E2 P ∧ Q <=> Q ∧ P } Commutative laws

E3 P V Q <=> Q V P

E4 (P ∧ Q) ∧ R <=> P ∧ (Q ∧ R) } Associative laws

E5 (P V Q) V R <=> PV (Q V R)

E6 P ∧ (Q V R) <=> (P ∧ Q) V (P ∧ R) } Distributive laws

E7 P V (Q ∧ R) <=> (P V Q) ∧ (PVR)

E8 7(P ∧ Q) <=> 7P V7Q

E9 7(P V Q) <=>7P ∧ 7Q } De Morgan's laws

E10 P V P <=> P

E11 P ∧ P <=> P

E12 R V (P ∧ 7P) <=>R

E13 R ∧ (P V 7P) <=>R

E14 R V (P V 7P) <=>T

E15 R ∧ (P ∧ 7P) <=>F

E16 P → Q <=> 7P V Q

E17 7 (P→ Q) <=> P ∧ 7Q

E18 P → Q <=> 7Q → 7P

E19 P → (Q → R) <=> (P ∧ Q) → R

E20 7(PD Q) <=> P D 7Q

E21 PDQ <=> (P → Q) ∧ (Q → P)

E22 (PDQ) <=> (P ∧ Q) V (7 P ∧ 7Q)

Example 1.Show that R is logically derived from P → Q, Q → R, and P

Solution.	{1}	(1)	P → Q	Rule P
	{2}	(2)	P	Rule P
	{1, 2}	(3)	Q	Rule (1), (2) and I11
	{4}	(4)	Q → R	Rule P
	{1, 2, 4}	(5)	R	Rule (3), (4) and I11.

Example 2.Show that S V R tautologically implied by (P V Q) ∧ (P → R) ∧ (Q → S).

Solution .	{1}	(1)	P V Q	Rule P
	{1}	(2)	7P → Q	T, (1), E1 and E16
	{3}	(3)	Q → S	P
	{1, 3}	(4)	7P → S	T, (2), (3), and I13
	{1, 3}	(5)	7S → P	T, (4), E13 and E1
	{6}	(6)	P → R	P
	{1, 3, 6}	(7)	7S → R	T, (5), (6), and I13
	{1, 3, 6)	(8)	S V R	T, (7), E16 and E1

Example 3. Show that 7Q, P→ Q => 7P

Solution .	{1}	(1)	P → Q	Rule P
	{1}	(2)	7P → 7Q	T, and E 18
	{3}	(3)	7Q	P
	{1, 3}	(4)	7P	T, (2), (3), and I11 .

Example 4 .Prove that R ∧ (P V Q) is a valid conclusion from the premises PVQ ,

Q → R, P → M and 7M.

Solution .	{1}	(1)	P → M	P
	{2}	(2)	7M	P
	{1, 2}	(3)	7P	T, (1), (2), and I12
	{4}	(4)	P V Q	P

{1, 2 , 4}	(5)	Q	T, (3), (4), and I10.
{6}	(6)	Q → R	P
{1, 2, 4, 6}	(7)	R	T, (5), (6) and I11
{1, 2, 4, 6}	(8)	R ∧ (PVQ)	T, (4), (7), and I9.

There is a third inference rule, known as rule CP or rule of *conditional proof*.

Rule CP: If we can derives s from R and a set of premises , then we can derive R → S from the set of premises alone.

Note. 1. Rule CP follows from the equivalence E10 which states that
(P ∧ R) → S óP → (R → S).
2. Let P denote the conjunction of the set of premises and let R be any formula The above equivalence states that if R is included as an additional premise and S is derived from P ∧ R then R → S can be derived from the premises P alone.
3. Rule CP is also called the *deduction theorem* and is generally used if the conclusion is of the form R → S. In such cases, R is taken as an additional premise and S is derived from the given premises and R.

Example 5 .Show that R → S can be derived from the premises
P → (Q → S), 7R V P , and Q.

Solution.

{1}	(1)	7R V P	P
{2}	(2)	R	P, assumed premise
{1, 2}	(3)	P	T, (1), (2), and I10
{4}	(4)	P → (Q → S)	P
{1, 2, 4}	(5)	Q → S	T, (3), (4), and I11
{6}	(6)	Q	P
{1, 2, 4, 6}	(7)	S	T, (5), (6), and I11
{1, 4, 6}	(8)	R → S	CP.

Example 6.Show that P → S can be derived from the premises, 7P V Q, 7Q V R, and R → S .

Solution.

{1}	(1)	7P V Q	P
{2}	(2)	P	P, assumed premise
{1, 2}	(3)	Q	T, (1), (2) and I11
{4}	(4)	7Q V R	P
{1, 2, 4}	(5)	R	T, (3), (4) and I11
{6}	(6)	R → S	P
{1, 2, 4, 6}	(7)	S	T, (5), (6) and I11
{2, 7}	(8)	P → S	CP

Example 7. " If there was a ball game , then traveling was difficult. If they arrived on time, then traveling was not difficult. They arrived on time. Therefore, there was no ball game". Show that these statements constitute a valid argument.

Solution. Let P: There was a ball game
Q: Traveling was difficult.
R: They arrived on time.

Given premises are: P → Q, R → 7Q and R conclusion is: 7P

{1}	(1)	P → Q	P
{2}	(2)	R → 7Q	P
{3}	(3)	R	P
{2, 3}	(4)	7Q	T, (2), (3), and I11
{1, 2, 3}	(5)	7P	T, (2), (4) and I12

Consistency of premises:

Consistency

A set of formulas H1, H2, …, Hm is said to be consistent if their conjunction has the truth value T for some assignment of the truth values to be atomic appearing in H1, H2, …, Hm.

Inconsistency

If for every assignment of the truth values to the atomic variables, at least one of the formulas H1, H2, … Hm is false, so that their conjunction is identically false, then the formulas H1, H2, …, Hm are called inconsistent.

A set of formulas H1, H2, …, Hm is inconsistent, if their conjunction implies a contradiction, that is H1∧ H2∧ … ∧ Hm => R ∧ 7R

Where R is any formula. Note that R ∧ 7R is a contradiction and it is necessary and sufficient that H1, H2, …,Hm are inconsistent the formula.

Indirect method of proof

In order to show that a conclusion C follows logically from the premises H1, H2,…, Hm, we assume that C is false and consider 7C as an additional premise. If the new set of premises is inconsistent, so that they imply a contradiction, then the assumption that 7C is true does not hold simultaneously with H1∧ H2∧ ..… ∧ Hm being true. Therefore, C is true whenever H1∧ H2∧ ..… ∧ Hm is true. Thus, C follows logically from the premises H1, H2 …..., Hm.

Example 8 Show that 7(P ∧ Q) follows from 7P∧ 7Q.

Solution.

We introduce 77 (P∧ Q) as an additional premise and show that this additional premise leads to a contradiction.

{1}	(1) 77(P∧ Q)	P assumed premise
{1}	(2) P∧ Q	T, (1) and E1
{1}	(3) P	T, (2) and I1

{1}	(4)	7P∧7Q	P
{4}	(5)	7P	T, (4) and I1
{1, 4}	(6)	P∧ 7P	T, (3), (5) and I9

Here (6) P∧ 7P is a contradiction. Thus {1, 4} viz. 77(P∧ Q) and 7P∧ 7Q leads to a contradiction P ∧ 7P.

Example 9Show that the following premises are inconsistent.

1. If Jack misses many classes through illness, then he fails high school.
2. If Jack fails high school, then he is uneducated.
3. If Jack reads a lot of books, then he is not uneducated.
4. Jack misses many classes through illness and reads a lot of books.

Solution.

P: Jack misses many classes.

Q: Jack fails high school.

R: Jack reads a lot of books.

S: Jack is uneducated.

The premises are P→ Q, Q → S, R→ 7S and P∧ R

{1}	(1)	P→Q	P
{2}	(2)	Q→ S	P
{1, 2}	(3)	P → S	T, (1), (2) and I13
{4}	(4)	R→ 7S	P
{4}	(5)	S → 7R	T, (4), and E18
{1, 2, 4}	(6)	P→7R	T, (3), (5) and I13
{1, 2, 4}	(7)	7PV7R	T, (6) and E16
{1, 2, 4}	(8)	7(P∧R)	T, (7) and E8
{9}	(9)	P∧ R	P
{1, 2, 4, 9)	(10)	(P∧ R) ∧ 7(P∧ R)	T, (8), (9) and I9

The rules above can be summed up in the following table. The "Tautology" column shows how to interpret the notation of a given rule.

Rule of inference	Tautology	Name
p $\therefore \overline{p \vee q}$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore \overline{p}$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore \overline{p \wedge q}$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
p $p \rightarrow q$ $\therefore \overline{q}$	$((p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \overline{\neg p}$	$((\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore \overline{p \rightarrow r}$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore \overline{q}$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism
$p \vee q$ $\neg p \vee r$ $\therefore \overline{q \vee r}$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

Proof of contradiction:

The "Proof by Contradiction" is also known as reductio ad absurdum, which is probably Latin for "reduce it to something absurd".

Here's the idea:

1. Assume that a given proposition is untrue.

2. Based on that assumption reach two conclusions that contradict each other.

This is based on a classical formal logic construction known as Modus Tollens: If P implies Q and Q is false, then P is false. In this case, Q is a proposition of the form (R and not R) which is always false. P is the negation of the fact that we are trying to prove and if the negation is not true then the original proposition must have been true. If computers are not "not stupid" then they are stupid. (I hear that "stupid computer!" phrase a lot around here.)

Example:

Lets prove that there is no largest prime number (this is the idea of Euclid's original proof). Prime numbers are integers with no exact integer divisors except 1 and themselves.

1. To prove: "There is no largest prime number" by contradiction.
2. Assume: There is a largest prime number, call it p.
3. Consider the number N that is one larger than the product of all of the primes smaller than or equal to p. N=1*2*3*5*7*11...*p + 1. Is it prime?
4. N is at least as big as p+1 and so is larger than p and so, by Step 2, cannot be prime.
5. On the other hand, N has no prime factors between 1 and p because they would all leave a remainder of 1. It has no prime factors larger than p because Step 2 says that there are no primes larger than p. So N has no prime factors and therefore must itself be prime (see note below).

We have reached a contradiction (N is not prime by Step 4, and N is prime by Step 5) and therefore our original assumption that there is a largest prime must be false.

Note: The conclusion in Step 5 makes implicit use of one other important theorem: The Fundamental Theorem of Arithmetic: Every integer can be uniquely represented as the product of primes. So if N had a composite (i.e. non-prime) factor, that factor would itself have prime factors which would also be factors of N.

Predicative logic:

A predicate or propositional function is a statement containing variables. For instance "x + 2 = 7", "X is American", "x < y", "p is a prime number" are predicates. The truth value of the predicate depends on the value assigned to its variables. For instance if we replace x with 1 in the predicate "x + 2 = 7" we obtain "1 + 2 = 7", which is false, but if we replace it with 5 we get "5 + 2 = 7", which is true. We represent a predicate by a letter followed by the variables enclosed between parenthesis: P (x), Q(x, y), etc. An example for P (x) is a value of x for which P (x) is true. A counterexample is a value of x for which P (x) is false. So, 5 is an example for "x + 2 = 7", while 1 is a counterexample. Each variable in a predicate is assumed to belong to a universe (or domain) of discourse, for instance in the predicate "n is an odd integer" 'n' represents an integer, so the universe of discourse of n is the set of all integers. In "X is American" we may assume that X is a human being, so in this case the universe of discourse is the set of all human beings.

Free & Bound variables:

Let's now turn to a rather important topic: the distinction between **free variables** and **bound variable**s.

Have a look at the following formula:

$$\neg(\text{THERAPIST}(x) \vee \forall x(\text{MORON}(x) \wedge \forall y\text{PERSON}(y)))$$

The first occurrence of x is *free*, whereas the second and third occurrences of x are *bound*, namely by the first occurrence of the quantifier ∀. The first and second occurrences of the variable y are also bound, namely by the second occurrence of the quantifier ∀.

Informally, the concept of a *bound variable* can be explained as follows: Recall that quantifications are generally of the form:

$\forall x\varphi$

or

$\exists x\varphi$

where $x$may be any variable. Generally, all occurences of this variable within the quantification are bound. But we have to distinguish two cases. Look at the following formula to see why:

$$\exists x(\text{MAN}(x) \wedge (\forall x\text{WALKS}(x)) \wedge \text{HAPPY}(x))$$

1. x may occur within another, embedded, quantification $\forall x\psi$ or $\exists x\psi$, such as the x in $\text{WALKS}(x)$ in our example. Then we say that it is bound by the quantifier of this embedded quantification (and so on, if there's another embedded quantification over x within ψ).
2. Otherwise, we say that it is bound by the top-level quantifier (like all other occurences of x in our example).

Here's a full formal simultaneous definition of *free* and *bound*:

1. Any occurrence of any variable is free in any atomic formula.
2. No occurrence of any variable is bound in any atomic formula.
3. If an occurrence of any variable is free in φ or in ψ, then that same occurrence is free in $\neg\varphi$, $(\varphi \to \psi)$, $(\varphi \vee \psi)$, and $(\varphi \wedge \psi)$.
4. If an occurrence of any variable is bound in φ or in ψ, then that same occurrence is bound in $\neg\varphi$, $(\varphi \to \psi)$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$. Moreover, that same occurrence is bound in $\forall y\varphi$ and $\exists y\varphi$ as well, for any choice of variable y.
5. In any formula of the form $\forall y\varphi$ or $\exists y\varphi$ (where y can be any variable at all in this case) the occurrence of y that immediately follows the initial quantifier symbol is bound.
6. If an occurrence of a variable x is free in φ, then that same occurrence is free in $\forall y\varphi$ and $\exists y\varphi$, for any variable y distinct from x. On the other hand, all occurrences of x that are free in φ, are bound in $\forall x\varphi$ and in $\exists x\varphi$.

   If a formula contains no occurrences of free variables we call it a ***sentence***

Quantifiers

The variable of predicates is quantified by quantifiers. There are two types of quantifier in predicate logic − Universal Quantifier and Existential Quantifier.

Universal Quantifier

Universal quantifier states that the statements within its scope are true for every value of the specific variable. It is denoted by the symbol ∀

$\forall x P(x)$ is read as for every value of x, P(x) is true..

Example − "Man is mortal" can be transformed into the propositional form ∀xP(x)∀xP(x) where P(x) is the predicate which denotes x is mortal and the universe of discourse is all men.

Existential Quantifier

Existential quantifier states that the statements within its scope are true for some values of the specific variable. It is denoted by the symbol ∃.

$\exists x P(x)$ is read as for some values of x, P(x) is true.

Example − "Some people are dishonest" can be transformed into the propositional form $\exists x P(x)$ where P(x) is the predicate which denotes x is dishonest and the universe of discourse is some people.

UNIT-III

Relations

Syllabus

Relations: Relations**,** Properties of binary Relations, Types of relations: equivalence, compatibility and partial ordering relations, Hasse diagram. Lattices and its properties.

Functions: introduction to Functions , types of functions.

Introduction

The elements of a set may be related to one another. For example, in the set of natural numbers there is the 'less than' relation between the elements. The elements of one set may also be related to the elements another set.

Binary Relation

A binary relation between two sets A and B is a rule R which decides, for any elements, whether a is in relation R to b. If so,we write a R b.If a is not in relation R to b,then we shall write a /R b.

We can also consider a R b as the ordered pair (a, b) in which case we can define a binary relation from A to B as a subset of A X B. This subset is denoted by the relation R.

In general, **any set of ordered pairs defines a binary relation.**

For example, the relation of father to his child is F = {(a, b) / a is the father of b}
In this relation F, the first member is the name of the father and the second is the name of the child.

The definition of relation permits any set of ordered pairs to define a relation.

For example, the set S given by

S = {(1, 2), (3, a), (b, a) ,(b, Joe)}

Definition

The **domain** D of a binary relation S is the set of all first elements of the ordered pairs in the relation.(i.e) D(S)= {a / $ b for which (a, b) Є S}

The **range** R of a binary relation S is the set of all second elements of the ordered pairs in the relation.(i.e) R(S) = {b / $ a for which (a, b) Є S}.

For example

For the relation S = {(1, 2), (3, a), (b, a) ,(b, Joe)}

D(S) = {1, 3, b, b} and

R(S) = {2, a, a, Joe}

Let X and Y be any two sets. A subset of the Cartesian product X * Y defines a relation, say C. For any such relation C, we have D(C) Í X and R(C) Í Y, and the relation C is said to from X to Y. If Y = X, then C is said to be a relation form X to X. In such case, c is called a relation in X. Thus any relation in X is a subset of X * X . The set X * X is called a *universal relation* in X, while the empty set which is also a subset of X * X is called a *void relation* in X.

For example

Let L denote the relation "less than or equal to" and D denote the relation "divides" where x D y means " x divides y". Both L and D are defined on the set {1, 2, 3, 4}

L = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)}

D = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)}

L Ç D = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)}

= D

Properties of Binary Relations:

1.reflexive

Definition: A binary relation R in a set X is **reflexive** if, for every x Є X, x R x,

That is (x, x) Є R, or R is reflexive in X ó (x) (x Є X ® x R x).

For example

- The relation £ is reflexive in the set of real numbers.
- The set inclusion is reflexive in the family of all subsets of a universal set.
- The relation equality of set is also reflexive.
- The relation is parallel in the set lines in a plane.

- The relation of similarity in the set of triangles in a plane is reflexive.

Examples: (i). If $R_1 = \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3)\}$ be a relation on $A = \{1, 2, 3\}$, then R_1 is

- a reflexive relation, since for every $x \in A$, $(x, x) \in R_1$.

(ii). If $R_2 = \{(1, 1), (1, 2), (2, 3), (3, 3)\}$ be a relation on $A = \{1, 2, 3\}$, then R_2 is not a reflexive relation, since for every $2 \in A$, $(2, 2) \notin R_2$.

Symmetric

Definition: A relation R in a set X is **symmetric** if for every x and y in X, whenever x R y, then y R x.(i.e) R is symmetric in X ó (x) (y) (x Є X ^ y Є X ^ x R y ® y R x}

For example

- The relation equality of set is symmetric.
- The relation of similarity in the set of triangles in a plane is symmetric.
- The relation of being a sister is not symmetric in the set of all people.
- However, in the set females it is symmetric.

Example. If $R_3 = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 1), (3, 1)\}$ be a relation on $A = \{1, 2, 3\}$, then R_3 is a symmetric relation.

Transitive

Definition: A relation R in a set X is **transitiv**e if, for every x, y, and z are in X, whenever x R y and y R z , then x R z. (i.e) R is transitive in X ó (x) (y) (z) (x Є X^ y Є X^ z Є X ^ x R y^ y R z® x R z)

For example

- The relations **<, £, >, ³** and **=** are transitive in the set of real numbers
- The relations Í, Ì, Ê, É and equality are also transitive in the family of sets.
- The relation of similarity in the set of triangles in a plane is transitive.
- **Definition:**A relation R in a set x is **anti symmetric** if , for every x and y in X, whenever x R y and y R x, then x = y. Symbolically,(x) (y) (x Є X ^ y Є X ^ x R y ^ y R x ® x = y)

Example:If $R_4 = \{(1, 2), (2, 2), (2, 3)\}$ on $A = \{1, 2, 3\}$ is an antisymmetric.

Equivalence Relation:

Definition:A relation R in a set A is called an **equivalence** relation if

- a R a for every i.e. R is reflexive
- a R b => b R a for every a, b Є A i.e. R is symmetric
- a R b and b R c => a R c for every a, b, c Є A, i.e. R is transitive.

For example

- The relation equality of numbers on set of real numbers.
- The relation being parallel on a set of lines in a plane.

Problem1: Let us consider the set T of triangles in a plane. Let us define a relation

R in T as R= {(a, b) / (a, b Є T and a is similar to b}

We have to show that relation R is an equivalence relation

Solution :

- A triangle a is similar to itself. a R a
- If the triangle a is similar to the triangle b, then triangle b is similar to the triangle a then a R b => b R a
- If a is similar to b and b is similar to c, then a is similar to c (i.e) a R b and b R c => a R c.

Hence R is an equivalence relation.

Problem 2: Let x = {1, 2, 3, … 7} and R = {(x, y) / x – y is divisible by 3}

Show that R is an equivalence relation.

Solution: For any a Є X, a- a is divisible by 3,

Hence a R a, R is reflexive

For any a, b Є X, if a – b is divisible by 3, then b – a is also divisible by 3,

R is symmetric.

For any a, b, c Є, if a R b and b R c, then a – b is divisible by 3 and

b–c is divisible by 3. So that (a – b) + (b – c) is also divisible by 3,

hence a – c is also divisible by 3. Thus R is transitive.

Hence R is equivalence.

Problem3 Let Z be the set of all integers. Let m be a fixed integer. Two integers a and b are said to be congruent modulo m if and only if m divides a-b, in which case we write a º b (mod m). This relation is called the relation of congruence modulo m and we can show that is an equivalence relation.

Solution :

- a - a=0 and m divides a – a (i.e) a R a, (a, a) Є R, R is reflexive .
- a R b = m divides a-b

m divides b - a

b º a (mod m)

b R a

that is R is symmetric.

- a R b and b R c => a ºb (mod m) and bº c (mod m)
    - m divides a – b and m divides b-c
    - a – b = km and b – c = lm for some k ,l Є z
    - (a – b) + (b – c) = km + lm
    - a – c = (k +l) m
    - aº c (mod m)
    - a R c
    - R is transitive

Hence the congruence relation is an equivalence relation.

Equivalence Classes:

Let R be an equivalence relation on a set A. For any a ЄA, the **equivalence class** generated by a is the set of all elements b Є A such a R b and is denoted [a]. It is also called the R – equivalence class and denoted by a Є A.

i.e., [a] = {b Є A / b R a}

Let Z be the set of integer and R be the relation called "congruence modulo 3"

defined by R = {(x, y)/ xÎ Z Ù yÎZ Ù (x-y) is divisible by 3}

Then the equivalence classes are

[0] = {… -6, -3, 0, 3, 6, …}

[1] = {…, -5, -2, 1, 4, 7, …}

[2] = {…, -4, -1, 2, 5, 8, …}

Composition of binary relations:

Definition: Let R be a relation from X to Y and S be a relation from Y to Z. Then the relation R o S is given by R o S = {(x, z) / xÎX Ù z Î Z Ù y Î Y such that (x, y) Î R Ù (y, z) Î S)} is called the composite relation of R and S.

The operation of obtaining R o S is called the **composition of relations**.

Example: Let R = {(1, 2), (3, 4), (2, 2)} and

S = {(4, 2), (2, 5), (3, 1),(1,3)}

Then R o S = {(1, 5), (3, 2), (2, 5)} and S o R = {(4, 2), (3, 2), (1, 4)}

It is to be noted that R o S ≠ S o R.

Also Ro(S o T) = (R o S) o T = R o S o T

Note: We write R o R as R2; R o R o R as R3 and so on.

Definition

Let R be a relation from X to Y, a relation R from Y to X is called the **converse** of R, where the ordered pairs of Ř are obtained by interchanging the numbers in each of the ordered pairs of R. This means for x Î X and y Î Y, that x R y ó y Ř x.

Then the relation Ř is given by R = {(x, y) / (y, x) Î R} is called the **converse** of R

Example:

Let R = {(1, 2),(3, 4),(2, 2)}

Then Ř = {(2, 1),(4, 3),(2, 2)}

Note: If R is an equivalence relation, then Ř is also an equivalence relation.

Partial Ordering Relations:

Definition

A binary relation *R* in a set *P* is called a *partial order relation* or a *partial ordering* in *P* if *R* isreflexive, ant symmetric, and transitive.
i.e.,

- aRa for all $a \in P$
- aRb and $bRa \Rightarrow a = b$
- aRb and $bRc \Rightarrow aRc$

A set *P* together with a partial ordering *R* is called a *partial ordered set* or *poset*. The relation *R* is often denoted by the symbol $\preceq$ which is different from the usual less than equal to symbol.

Thus, if $\preceq$ is a partial order in *P* , then the ordered pair $(P, \preceq)$ is called a poset.

Example: Show that the relation ‖greater than or equal to‖ is a partial ordering on the set of integers.

Solution: Let *Z* be the set of all integers and the relation $R = \geq$

(i). Since $a \geq a$ for every integer *a*, the relation $\geq$ is reflexive.

(ii). Let *a* and *b* be any two integers.

Let aRb and $bRa \Rightarrow a \geq b$ and $b \geq a$

$\Rightarrow a = b$

$\therefore$ The relation $\geq$ is antisymmetric.

(iii).Let *a, b* and *c* be any three integers.

Let aRb and $bRc \Rightarrow a \geq b$ and $b \geq c$

$\Rightarrow a \geq c$

$\therefore$ The relation Let aRb and $bRc \Rightarrow a \geq b$ and $b \geq c$

$\Rightarrow a \geq c$

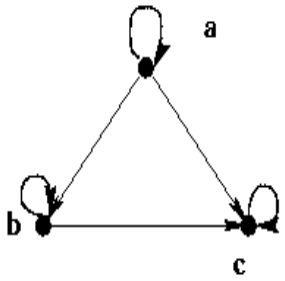
$\therefore$ The relation $\geq$ is transitive.

Since the relation $\geq$ is reflexive, antisymmetric and transitive, $\geq$ is partial ordering on the set of integers. Therefore, $(Z, \geq)$ is a poset

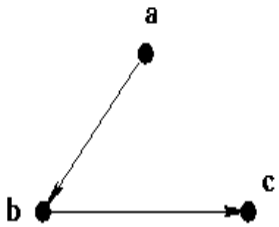
Hasse Diagram:

A Hasse diagram is a digraph for a poset which does not have loops and arcs implied by the transitivity.

Example 10: For the relation {< a, a >, < a, b >, < a, c >, < b, b >, < b, c >, < c, c >} on set {a, b,c}, the Hasse diagram has the arcs {< a, b >, < b, c >} as shown below.

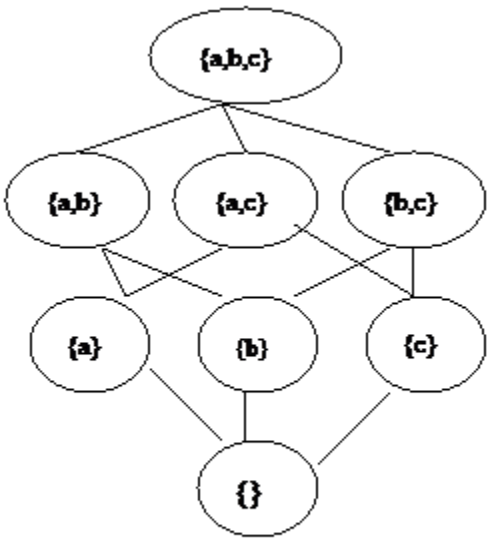


Digraph for Partial Order



Hasse Diagram

Ex: Let A be a given finite set and r(A) its power set. Let Í be the subset relation on the elements of r(A). Draw Hasse diagram of (r(A), Í) for A = {a, b, c}



Functions:

Introduction

A function is a special type of relation. It may be considered as a relation in which each element of the domain belongs to only one ordered pair in the relation. Thus a function from A to B is a subset of A X B having the property that for each a ЄA, there is one and only one b Є B such that (a, b) Î G.

Definition

Let A and B be any two sets. A relation f from A to B is called a function if for every a Є A there is a unique b Є B such that (a, b) Є f .

Note that the definition of function requires that a relation must satisfy two additional conditions in order to qualify as a function.

The first condition is that every a Є A must be related to some b Є B, (i.e) the domain of f must be A and not merely subset of A. The second requirement of uniqueness can be expressed as (a, b) Є f ^ (b, c) Є f => b = c

Intuitively, a function from a set A to a set B is a rule which assigns to every element of A, a unique element of B. If a ЄA, then the unique element of B assigned to a under f is denoted by f (a).The usual notation for a function f from A to B is f: A® B defined by a ® f (a) where a Є A, f(a) is called the image of **a** under f and **a** is called pre image of f(a).

- Let X = Y = **R** and f(x) = x2 + 2. Df = R and Rf Í R.
- Let X be the set of all statements in logic and let Y = {True, False}.

A mapping f: X®Y is a function.

- A program written in high level language is mapped into a machine language by a compiler. Similarly, the output from a compiler is a function of its input.
- Let X = Y = **R** and f(x) = x2 is a function from X ® Y,and g(x2) = x is not a function from X ® Y.

A mapping f: A ® B is called **one-to-one** (injective or 1 –1) if distinct elements of A are mapped into distinct elements of B. (i.e) f is one-to-one if

a1 = a2 => f (a1) = f(a2) or equivalently f(a1) ¹ f(a2) => a1 ¹ a2

For example, f: N ® N given by f(x) = x is 1-1 where N is the set of a natural numbers.

A mapping f: A® B is called **onto** (surjective) if for every b Є B there is an a Є A such that f (a) = B. i.e. if every element of B has a pre-image in A. Otherwise it is called **into**.

For example, f: Z®Z given by f(x) = x + 1 is an onto mapping.

A mapping is both 1-1 and onto is called bijective

.

For example f: R®R given by f(x) = X + 1 is bijective.

Definition: A mapping f: R® b is called a **constant mapping** if, for all aÎA, f (a) = b,

a fixed element.

For example f: Z®Z given by f(x) = 0, for all x ÎZ is a constant mapping.

Definition

A mapping f: A®A is called the **identity mapping** of A if f (a) = a,

for all aÎA. Usually it is denoted by IA or simply I.

Composition of functions:

If f: A®B and g: B®C are two functions, then the composition of functions f and g, denoted by g o f, is the function is given by g o f : A®C and is given by

g o f = {(a, c) / a Є A ٨ c Є C ٨ $bÎ B ': f(a)= b ٨ g(b) = c}

and (g of)(a) = ((f(a))

Example 1: Consider the sets A = {1, 2, 3},B={a, b} and C = {x, y}.

Let f: A® B be defined by f (1) = a ; f(2) = b and f(3)=b and

Let g: B® C be defined by g(a) = x and g(b) = y

(i.e) f = {(1, a), (2, b), (3, b)} and g = {(a, x), (b, y)}.

Then g o f: A®C is defined by

(g of) (1) = g (f(1)) = g(a) = x

(g o f) (2) = g (f(2)) = g(b) = y

(g o f) (3) = g (f(3)) = g(b) = y

i.e., g o f = {(1, x), (2, y),(3, y)}

If f: A® A and g: A®A, where A= {1, 2, 3}, are given by

f = {(1, 2), (2, 3), (3, 1)} and g = {(1, 3), (2, 2), (3, 1)}

Then g of = {(1, 2), (2, 1), (3, 3)}, fog= {(1, 1), (2, 3), (3, 2)}

f of = {(1, 3), (2, 1), (3, 2)} and gog= {(1, 1), (2, 2), (3, 3)}

Example 2: Let f(x) = x+2, g(x) = x – 2 and h(x) = 3x for x Î R, where R is the set of real numbers.

Then f o f = {(x, x+4)/xÎ R}

f o g = {(x, x)/ x Î X}

g o f = {(x, x)/ xÎ X}

g o g = {(x, x-4)/x Î X}

h o g = {(x,3x-6)/ x Î X}

h o f = {(x, 3x+6)/ x Î X}

Inverse functions:

Let f: A® B be a one-to-one and onto mapping. Then, its inverse, denoted by f -1 is given by f -1 = {(b, a) / (a, b) Î f} Clearly f-1: B® A is one-to-one and onto.

Also we observe that f o f -1 = IB and f -1o f = IA.

If f -1 exists then f is called **invertible**.

For example:Let f: R ®R be defined by f(x) = x + 2

Then f -1: R® R is defined by f -1(x) = x - 2

Theorem: Let f: X ®Y and g: Y ® Z be two one to one and onto functions. Then gof is also one to one and onto function.

Proof

Let f:X ® Y g : Y ® Z be two one to one and onto functions. Let x1, x2 Î X

- g o f (x1) = g o f(x2),
- g (f(x1)) = g(f(x2)),
- g(x1) = g(x2) since [f is 1-1]

x1 = x2 since [g is 1-1}

so that gof is 1-1.

By the definition of composition, gof : X ® Z is a function.

We have to prove that every element of z Î Z an image element for some x Î X under gof.

Since g is onto $ y ÎY ': g(y) = z and f is onto from X to Y,

$ x ÎX ': f(x) = y.

Now, gof (x) = g (f (x))

= g(y) [since f(x) = y]

= z [since g(y) = z]

which shows that gof is onto.

Theorem (g o f) -1 = f -1 o g -1

(i.e) the inverse of a composite function can be expressed in terms of the composition of the inverses in the reverse order.

Proof.

f: A ® B is one to one and onto.

g: B ® C is one to one and onto.

gof: A ® C is also one to one and onto.

Þ (gof) -1: C ® A is one to one and onto.

Let a Î A, then there exists an element b Î b such that f (a) = b Þ a = f-1 (b).

Now b Î B Þ there exists an element c Î C such that g (b) = c Þ b = g -1(c).

Then (gof)(a) = g[f(a)] = g(b) = c Þ a = (gof) -1(c). …….(1)

(f -1 o g-1) (c) = f -1(g -1 (c)) = f -1(b) = a Þ a = (f -1 o g -1)(c) ….(2)

Combining (1) and (2), we have

(gof) -1 = f -1 o g -1

Theorem: If f: A ® B is an invertible mapping , then

f o f -1 = I B and f-1 o f = IA

Proof: f is invertible, then f -1 is defined by f(a) = b ó f-1(b) = a

where a Î A and bÎ B .

Now we have to prove that f of -1 = IB .

Let bÎ B and f -1(b) = a, a Î A

then fof-1(b) = f(f-1(b))

= f(a) = b

therefore f o f -1 (b) = b " b Î B => f o f -1 = IB

Now f -1 o f(a) = f -1 (f(a)) = f -1 (b) = a

therefore f -1 o f(a) = a " a Î A => f -1 o f = IA.

Lattice and its Properties:

Introduction:

A lattice is partially ordered set (L, £) in which every pair of elements a, b ÎL has a greatest lower bound and a least upper bound.

The glb of a subset, {a, b} Í L will be denoted by a * b and the lub by a Å b.

.

Usually, for any pair a, b Î L, GLB {a, b} = a * b, is called the **meet** or **product** and LUB{a, b} = a Å b, is called the **join** or **sum** of a and b.

Example1 Consider a non-empty set S and let P(S) be its power set. The relation Í "contained in" is a partial ordering on P(S). For any two subsets A, BÎ P(S)

GLB {A, B} and LUB {A, B} are evidently A Ç B and A È B respectively.

Example2 Let I+ be the set of positive integers, and D denote the relation of "division" in I+ such that for any a, b Î I+ , a D b iff a divides b. Then (I+, D) is a lattice in which the join of a and b is given by the least common multiple(LCM) of a and b, that is, a Å b = LCM of a and b, and the meet of a and b, that is , a * b is the greatest common divisor (GCD) of a and b.

A lattice can be conveniently represented by a diagram.

For example, let Sn be the set of all divisors of n, where n is a positive integer. Let D denote the relation "division" such that for any a, b Î Sn, a D b iff a divides b.

Then (Sn, D) is a lattice with a * b = gcd(a, b) and a Å b = lcm(a, b).

Take n=6. Then S6 = {1, 2, 3, 6}. It can be represented by a diagram in Fig(1).

Take n=8. Then S8 = {1, 2, 4, 8}

Two lattices can have the same diagram. For example if S = {1, 2, 3} then (p(s), Í) and (S6,D) have the same diagram viz. fig(1), but the nodes are differently labeled .

We observe that for any partial ordering relation £ on a set S the converse relation ³ is also partial ordering relation on S. If (S, £) is a lattice With meet a * b and join a Å b , then (S, ³) is the also a lattice with meet a Å b and join a * b i.e., the GLB and LUB get interchanged . Thus we have the principle of duality of lattice as follows.

Any statement about lattices involving the operations ^ and V and the relations £ and ³ remains true if ^, V, ³ and £ are replaced by V, ^, £ and ³ respectively.

The operation ^ and V are called duals of each other as are the relations £ and ³.. Also, the lattice (L, £) and (L, ³) are called the duals of each other.

Properties of lattices:

Let (L, £) be a lattice with the binary operations * and Å then for any a, b, c Î L,

- a * a = a a Å a = a (Idempotent)
- a * b = b * a , a Å b = b Å a (Commutative)
- (a * b) * c = a * (b * c) , (a Å) Å c = a Å (b Å c)

- (Associative)

- a * (a Å b) = a , a Å (a * b) = a (absorption)

For any a ÎL, a £ a, a £ LUB {a, b} => a £ a * (a Å b). On the other hand,
GLB {a, a Å b} £ a i.e., (a Å b) Å a, hence a * (a Å b) = a

Theorem 1

Let (L, £) be a lattice with the binary operations * and Å denote the operations of meet and join respectively For any a, b Î L,

a £ b ó a * b = a ó a Å b = b

Proof

Suppose that a £ b. we know that a £ a, a £ GLB {a, b}, i.e., a £ a * b.
But from the definition of a * b, we get a * b £ a.
Hence a £ b => a * b = a …… (1)
Now we assume that a * b = a; but is possible only if a £ b,
that is a * b = a => a £ b …… (2)
From (1) and (2), we get a £ b ó a * b = a.
Suppose a * b = a.
then b Å (a * b) = b Å a = a Å b ……. (3)
but b Å (a * b) = b (by iv) …….. (4)
Hence a Å b = b, from (3) => (4)
Suppose aÅ b = b, i.e., LUB {a, b} = b, this is possible only if a£ b, thus(3) => (1)
(1) => (2) => (3) => (1). Hence these are equivalent.

Let us assume a * b = a.
Now (a * b) Å b = a Å b
We know that by absorption law , (a * b) Å b = b
so that a Å b = b, therefore a * b = a Þ a Å b = b (5)
similarly, we can prove a Å b = b Þ a * b = a (6)
From (5) and (6), we get

a * b = a Û a Å b = b

Hence the theorem.

Theorem2 For any a, b, c Î L, where (L, £) is a lattice.

b £ c => { a * b £ a * c and

{ a Å b £ a Å c

Proof Suppose b £ c. we have proved that b £ a ó b * c = b…….. (1)

Now consider

(a * b) * (a * c) = (a * a) * (b * c) (by Idempotent)

= a * (b * c)

= a * b (by (1))

Thus (a * b) * (a * c) = a * b which => (a * b) £ (a * c)

Similarly (a Å b) Å (a Å c) = (a Å a) Å (b Å c)

= a Å (b Å c)

= a Å c

which => (a Å b) £ (a Å c)

note:These properties are known as **isotonicity.**

Unit-IV:
Algebraic Structures

Syllabus:

Algebraic structures: Algebraic systems with examples and general properties, semi groups and monoids, groups & its types, Introduction to homomorphism and Isomorphism (Proof of theorems are not required)

Algebraic systems

N = {1,2,3,4,….. } = Set of all natural numbers.

Z = { 0, ± 1, ± 2, ± 3, ± 4 , ….. } = Set of all integers. Q = Set of all rational numbers.

R = Set of all real numbers.

Binary Operation: The binary operator * is said to be a binary operation (closed operation) on a non- empty set A, if $a * b \in A$ for all $a, b \in A$ (Closure property).

Ex: The set N is closed with respect to addition and multiplication but not w.r.t subtraction and division.

Algebraic System: A set A with one or more binary(closed) operations defined on it is called an algebraic system.

Ex: (N, +), (Z, +, –), (R, +, . , –) are algebraic systems.

Properties

Associativity: Let * be a binary operation on a set A.

The operation * is said to be associative in A .

if $(a * b) * c = a *(b* c)$ for all a, b, c in A

Identity: For an algebraic system (A, *), an element 'e' in A is said to be an identity element of A if a * e = e * a = a for all a ∈ A.

Note: For an algebraic system (A, *), the identity element, if exists, is unique.

Inverse: Let (A, *) be an algebraic system with identity 'e'. Let a be an element in A. An element b is said to be inverse of A .

if a * b = b * a = e

Semi groups

Semi Group: An algebraic system (A, *) is said to be a semi group if

1. * is closed operation on A.
2. * is an associative operation, for all a, b, c in

A. Ex. (N, +) is a semi group.

Ex. (N, .) is a semi group.

Ex. (N, –) is not a semi group.

Monoid

An algebraic system (A, *) is said to be a **monoid** if the following conditions are satisfied.

1) * is a closed operation in A.
2) * is an associative operation in A.
3) There is an identity in A.

Ex. Show that the set 'N' is a monoid with respect to multiplication. Solution: Here, N = {1,2,3,4,……}

Closure property: We know that product of two natural numbers is again a natural number.

i.e., $a.b = b.a$ for all $a,b \in N$

∴ Multiplication is a closed operation.

Associativity : Multiplication of natural numbers is associative.

i.e., $(a.b).c = a.(b.c)$ for all $a,b,c \in N$

Identity : We have, $1 \in N$ such that

$a.1 = 1.a = a$ for all $a \in N$.

$\therefore$ Identity element exists, and 1 is the identity element.

Hence, N is a monoid with respect to multiplication

Examples

Ex. Let (Z, *) be an algebraic structure, where Z is the set of integers

and the operation * is defined by $n * m$ = maximum of (n, m).

Show that (Z, *) is a semi group.

Is (Z, *) a monoid ?. Justify your answer.

Solution: Let a , b and c are any three integers.

Closure property: Now, $a * b =$ maximum of $(a, b) \in Z$ for all $a,b \in Z$

Associativity : $(a * b) * c$ = maximum of {a,b,c} = $a * (b * c)$

$\therefore$ (Z, *) is a semi group.

Identity : There is no integer x such that

$a * x$ = maximum of $(a, x) = a$ for all $a \in Z$

$\therefore$ Identity element does not exist. Hence, (Z, *) is not a monoid.

Ex. Show that the set of all strings 'S' is a monoid under the operation 'concatenation of strings'.

Is S a group w.r.t the above operation? Justify your answer.

Solution: Let us denote the operation

'concatenation of strings' by +.

Let s_1, s_2, s_3 are three arbitrary strings in S.

Closure property: Concatenation of two strings is again a string. i.e.,

$s_1+s_2 \in S$

Associativity: Concatenation of strings is associative.

$(s_1+ s_2) + s_3 = s_1+ (s_2 + s_3)$

Identity: We have null string , $l \in S$ such that $s_1 + l = S$.

$\therefore$ S is a monoid.

Note: S is not a group, because the inverse of a non empty string does not exist under concatenation of strings.

3.2 Groups

Group: An algebraic system (G, *) is said to be a **group** if the following conditions are satisfied.

1) * is a closed operation.

2) * is an associative operation.

3) There is an identity in G.

4) Every element in G has inverse in G.

Abelian group (Commutative group): A group (G, *) is said to be ***abelian*** (or ***commutative)*** if

$a * b = b * a \quad \forall a, b \in G.$

Properties

In a Group (G, *) the following properties hold good

1. Identity element is unique.
2. Inverse of an element is unique.
3. Cancellation laws hold good

$a * b = a * c \Rightarrow b = c$ (left cancellation law)

$a * c = b * c \Rightarrow a = b$ (Right cancellation law) 4. $(a * b)^{-1} = b^{-1} * a^{-1}$

In a group, the identity element is its own inverse.

Order of a group : The number of elements in a group is called order of the group.

Finite group: If the order of a group G is finite, then G is called a finite group.

Ex1 . Show that, the set of all integers is an abelian group with respect to addition.

Solution: Let Z = set of all integers.

Let a, b, c are any three elements of Z.

1. Closure property: We know that, Sum of two integers is again an integer.

   i.e., $a + b \in Z$ for all $a,b \in Z$

2. Associativity: We know that addition of integers is

associative. i.e., $(a+b)+c = a+(b+c)$ for all $a,b,c \in Z$.

3. Identity : We have $0 \in Z$ and $a + 0 = a$ for all $a \in Z$.

$\therefore$ Identity element exists, and '0' is the identity element.

4. Inverse: To each $a \in Z$, we have $-a \in Z$ such that

$$a + (- a) = 0$$

Each element in Z has an inverse.

5. Commutativity: We know that addition of integers is

commutative. i.e., $a + b = b + a$ for all $a,b \in Z$.

Hence, (Z , +) is an abelian group.

Ex2 . Show that set of all non zero real numbers is a group with respect to multiplication .

Solution: Let R^* = set of all non zero real numbers.

Let a, b, c are any three elements of R^* .

1. Closure property : We know that, product of two nonzero real numbers is again a nonzero real number .

i.e., $a \cdot b \in R^*$ for all $a,b \in R^*$.

2. Associativity: We know that multiplication of real numbers is

associative.

i.e., $(a.b).c = a.(b.c)$ for all $a,b,c \in R^*$.

3. Identity : We have $1 \in R^*$ and $a \cdot 1 = a$ for all $a \in R^*$.

$\therefore$ Identity element exists, and '1' is the identity element.

4. Inverse: To each $a \in R^*$, we have $1/a \in R^*$ such that

$a \cdot (1/a) = 1$ i.e., Each element in R^* has an inverse.

5. Commutativity: We know that multiplication of real numbers

is

commutative.

i.e., $a \cdot b = b \cdot a$ for all $a,b \in R^*$.

Hence, $(R^*, .)$ is an abelian group.

Note: Show that set of all real numbers 'R' is not a group with respect to multiplication.

Solution: We have $0 \in R$.

The multiplicative inverse of 0 does not

exist. Hence. R is not a group.

Example: Let S be a finite set, and let F(S) be the collection of all functions $f: S \to S$ under the operation of composition of functions, then show that F(S) is a monoid.

Is S a group w.r.t the above operation? Justify your answer.

Solution:

Let f_1, f_2, f_3 are three arbitrary functions on S.

Closure property: Composition of two functions on S is again a function on S. i.e., $f_1 o f_2 \in F(S)$

Associativity: Composition of functions is associative.

i.e., $(f_1 \text{ o } f_2) \text{ o } f_3 = f_1 \text{ o } (f_2 \text{ o } f_3)$

Identity: We have identity function $I : S \rightarrow S$

such that $f_1 \text{ o } I = f_1$.

$\therefore$ F(S) is a monoid.

Note: F(S) is not a group, because the inverse of a non bijective function on S does not exist.

Ex. If M is set of all non singular matrices of order 'n x n'. then show that M is a group w.r.t. matrix multiplication. Is (M, *) an abelian group?. Justify your answer.

Solution: Let A,B,C $\in$ M.

1. Closure property : Product of two non singular matrices is again a non singular matrix, because

½AB½=½A½.½B½¹0 (Since, A and B are nonsingular) i.e.,

AB $\in$ M for all A,B $\in$ M .

2. Associativity: Marix multiplication is associative.

i.e., (AB)C=A(BC) for all A,B,C $\in$ M .

3. Identity : We have $I_n \in M$ and $A\, I_n = A$ for all $A \in M$.

$\therefore$ Identity element exists, and 'I_n' is the identity element.

4. Inverse: To each $A \in M$, we have $A^{-1} \in M$ such that

$A\, A^{-1} = I_n$ i.e., Each element in M has an inverse.

$\therefore$ M is a group w.r.t. matrix multiplication.

We know that, matrix multiplication is not commutative.

Hence, M is not an abelian group.

Ex. Show that the set of all positive rational numbers forms an abelian group under the composition * defined by

a * b = (ab)/2 .

Solution: Let A = set of all positive rational numbers.

Let a,b,c be any three elements of A.

1. Closure property: We know that, Product of two positive rational numbers is again a

rational number.

i.e., a *b ∈ A for all a,b ∈ A .

2. Associativity: (a*b)*c = (ab/2) * c = (abc) / 4

a*(b*c) = a * (bc/2) = (abc) / 4

3. Identity : Let e be the identity element.

We have a*e = (ae)/2 …(1) , By the definition of *

again, a*e = a …..(2), Since e is the identity.

From (1) and (2), (ae)/2 = a ⇒ e = 2 and 2 ∈ A .

∴ Identity element exists, and '2' is the identity element in A.

4. Inverse: Let a ∈ A

let us suppose b is inverse of a. Now, a * b = (ab)/2 ….(1)
(By definition of inverse.)
Again, a * b = e = 2 …..(2)
(By definition of inverse)

From (1) and (2), it follows that

(a b)/2 = 2

=> b = (4 / a) ∈ A

∴ (A ,*) is a group.

Commutativity: a * b = (ab/2) = (ba/2) = b * a

Hence, (A,*) is an abelian group.

Finite groups

Ex. Show that G = {1, -1} is an abelian group under multiplication.

Solution: The composition table of G is

*	1	-1
1	1	-1
-1	-1	1

1. Closure property: Since all the entries of the composition table are the elements of the given set, the set G is closed under multiplication.

2. Associativity: The elements of G are real numbers, and we know that multiplication of real numbers is associative.

3. Identity : Here, 1 is the identity element and $1 \in G$.

4. Inverse: From the composition table, we see that the inverse elements of

1 and -1 are 1 and -1 respectively.

Hence, G is a group w.r.t multiplication.

5. Commutativity: The corresponding rows and columns of the table are identical. Therefore the binary operation . is commutative.

Hence, G is an abelian group w.r.t. multiplication..

Ex. Show that $G = \{1, w, w^2\}$ is an abelian group under multiplication.
Where 1, w, w^2 are cube roots of unity.

Solution: The composition table of G is

*	1	w	w^2
1	1	w	w^2
w	w	w^2	1
w^2	w^2	1	w

1. Closure property: Since all the entries of the composition table are the elements of the given set, the set G is closed under multiplication.

2. Associativity: The elements of G are complex numbers, and we know that multiplication of complex numbersis associative.

3. Identity : Here, 1 is the identity element and $1 \in G$.

4. Inverse: From the composition table, we see that the inverse elements of 1 w, w^2

are 1, w^2, w respectively.

Hence, G is a group w.r.t multiplication.

5. Commutativity: The corresponding rows and columns of the table are identical. Therefore the binary operation . is commutative.

Hence, G is an abelian group w.r.t. multiplication.

Modulo systems.

Addition modulo m ($+_m$)

let m is a positive integer. For any two positive integers a and b

$a +_m b = a + b$ if $a + b < m$

$a +_m b = r$ if $a + b \geq m$ where r is the remainder obtained by dividing (a+b) with m.

Multiplication modulo p (*m)

let p is a positive integer. For any two positive integers a and b $a *_m b = a b$ if $a b < p$

$a *_m b = r$ if $a b \geq p$ where r is the remainder obtained by dividing (ab) with p.

Ex. $3 *_5 4 = 2$, $5 *_5 4 = 0$, $2 *_5 2 = 4$

Example : The set G = {0,1,2,3,4,5} is a group with respect to addition modulo 6.

Solution: The composition table of G is

$+_6$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

1. Closure property: Since all the entries of the composition table are the elements of the given set, the set G is closed under $+_6$.

2. Associativity: The binary operation $+_6$ is associative in G.

for ex. $(2 +_6 3) +_6 4 = 5 +_6 4 = 3$ and

$2 +_6 (3 +_6 4) = 2 +_6 1 = 3$

3. Identity : Here, The first row of the table coincides with the top row. The element heading that row , i.e., 0 is the identity element.

4. . Inverse: From the composition table, we see that the inverse elements of 0, 1, 2, 3, 4. 5 are 0, 5, 4, 3, 2, 1 respectively

5.Commutativity: The corresponding rows and columns of the table are identical. Therefore the binary operation $+_6$ is commutative.

Hence, $(G, +_6)$ is an abelian group.

Example : The set G = {1,2,3,4,5,6} is a group with respect to multiplication modulo 7.

Solution: The composition table of G is

$*_7$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

1. Closure property: Since all the entries of the composition table are the elements of the given set, the set G is closed under $*_7$.

2. Associativity: The binary operation $*_7$ is associative in G.

for ex. $(2 *_7 3) *_7 4 = 6 *_7 4 = 3$ and

$2 *_7 (3 *_7 4) = 2 *_7 5 = 3$

3. Identity : Here, The first row of the table coincides with the top row. The element heading that row , i.e., 1 is the identity element.

4. . Inverse: From the composition table, we see that the inverse elements of 1, 2, 3, 4. 5 6 are 1, 4, 5, 2, 5, 6 respectively.

5. Commutativity: The corresponding rows and columns of the table are identical. Therefore the binary operation $*_7$ is commutative.

Hence, $(G, *_7)$ is an abelian group.

More on finitegroups

In a group with 2 elements, each element is its own inverse

In a group of even order there will be at least one element (other than identity element) which is its own inverse

The set G = {0,1,2,3,4,…..m-1} is a group with respect to addition modulo m.

The set G = {1,2,3,4,….p-1} is a group with respect to multiplication modulo p, where p is a prime number.

Order of an element of a group:

Let (G, *****) be a group. Let 'a' be an element of G. The smallest integer n such that $a^n = e$ is called order of 'a'. If no such number exists then the order is infinite.

Ex. G = {1, -1, i, -i} is a group w.r.t multiplication. The order –i is a) 2 b) 3 c) 4 d) 1

Ex. Which of the following is not true.

a) The order of every element of a finite group is finite and is a divisor of the order of the group.

b) The order of an element of a group is same as that of its inverse.

c) In the additive group of integers the order of every element except 0 is infinite

d) In the infinite multiplicative group of nonzero rational numbers the order of every element except 1 is infinite.

Ans. D

3.3 Sub groups

Def. A non empty sub set H of a group (G, *) is a sub group of G, if (H, *) is a group.

Note: For any group {G, *}, {e, * } and (G, *) are trivial sub groups.

Ex. G = {1, -1, i, -i } is a group w.r.t multiplication.

H_1 = { 1, -1 } is a subgroup of G .

H_2 = { 1 } is a trivial subgroup of G.

Ex. (Z , +) and (Q , +) are sub groups of the group (R +).

Theorem: A non empty sub set H of a group (G, *) is a sub group of G iff

i) $a * b \in H \quad \forall a, b \in H$

ii) $a^{-1} \in H \quad \forall a \in H$

Theorem

Homomorphism and Isomorphism.

Homomorphism : Consider the groups (G, *) and (G^1, $\oplus$)

A function f : G $\rightarrow$ G^1 is called a homomorphism if

f (a * b) = f(a) $\oplus$f (b)

Isomorphism: If a homomorphism f: G $\rightarrow$ G^1 is a bijection then f is called isomorphism between G and G^1 .

Then we write G $\equiv$ G^1

Example : Let R be a group of all real numbers under addition and R^+ be a group of all positive real numbers under multiplication. Show that the mapping $f : R \to R^+$ defined by $f(x) = 2^x$ for all $x \in R$ is an isomorphism.

Solution: First, let us show that f is a homomorphism. Let

$a , b \in R$.

Now, $f(a+b) = 2^{a+b}$

$= 2^a \, 2^b$

$= f(a).f(b)$

$\therefore$ f is an homomorphism.

Next, let us prove that f is a Bijection.

For any $a , b \in R$, Let, $f(a) = f(b)$

$\Rightarrow 2^a = 2^b$

$\Rightarrow a = b$

$\therefore$ f is one.to-one.

Next, take any $c \in R^+$.

Then $\log_2 c \in R$ and $f(\log_2 c) = 2^{\log_2 c} = c$.

$\Rightarrow$ Every element in R^+ has a pre image in R. i.e.,

f is onto.

$\therefore$ f is a bijection.

Hence, f is an isomorphism.

Ex. Let R be a group of all real numbers under addition and R^+ be a group of all positive real numbers under multiplication. Show that the mapping $f : R^+ \to R$ defined by $f(x) = \log_{10} x$ for all $x \in R$ is an isomorphism.

Solution: First, let us show that f is a homomorphism. Let

$a , b \in R^+$.

Now, $f(a.b) = \log_{10} (a.b)$

$= \log_{10} a + \log_{10} b$

$= f(a) + f(b)$

∴ f is an homomorphism.

Next, let us prove that f is a

Bijection. For any a , b $\in R^+$,

Let, f(a) = f(b)

$$\Rightarrow \log_{10} a = \log_{10} b$$

$$\Rightarrow a = b$$

$\therefore$ f is one.to-one.

Next, take any c $\in$ R.

Then $10^c \in R$ and $f(10^c) = \log_{10} 10^c = c$.

$\Rightarrow$ Every element in R has a pre image in

R^+. i.e., f is onto.

$\therefore$ f is a bijection.

Hence, f is an isomorphism.

UNIT-V

Graph Theory

Syllabus

Graph Theory: Representation of Graph, DFS, BFS, Spanning Trees, planar Graphs.

Representation of Graphs:

There are two different sequential representations of a graph. They are

- Adjacency Matrix representation
- Path Matrix representation

Adjacency Matrix Representation

Suppose G is a simple directed graph with m nodes, and suppose the nodes of G have been ordered and are called v1, v2, . . . , vm. Then the adjacency matrix A = (aij) of the graph G is the m x m matrix defined as follows:

aij = 1 if vi is adjacent to Vj, that is, if there is an edge (Vi, Vj)

0 otherwise

Suppose G is an undirected graph. Then the adjacency matrix A of G will be a symmetric matrix, i.e., one in which aij = aji; for every i and j.

Drawbacks

1. It may be difficult to insert and delete nodes in G.

2. If the number of edges is 0(m) or 0(m log2 m), then the matrix A will be sparse, hence a great deal of space will be wasted.

Path Matrix Represenation

Let G be a simple directed graph with m nodes, v1,v2, . . . ,vm. The path matrix of G is the m-square matrix P = (pij) defined as follows:

$$P_{ij} = \begin{cases} 1 & \text{if there is a path from } V_i \text{ to } V_j \\ 0 & \text{otherwise} \end{cases}$$

Graphs and Multigraphs

A graph G consists of two things:

1.A set V of elements called nodes (or points or vertices)

2.A set E of edges such that each edge e in E is identified with a unique

(unordered) pair [u, v] of nodes in V, denoted by e = [u, v]

Sometimes we indicate the parts of a graph by writing G = (V, E).

Suppose e = [u, v]. Then the nodes u and v are called the endpoints of e, and u and v are said to be adjacent nodes or neighbors. The degree of a node u, written deg(u), is the number of edges containing u. If deg(u) = 0 — that is, if u does not belong to any edge—then u is called an isolated node.

Path and Cycle

A path P of length n from a node u to a node v is defined as a sequence of n + 1 nodes. P = (v0, v1, v2, . . . , vn) such that u = v0; vi-1 is adjacent to vi for i = 1,2, . . ., n and vn = v.

Types of Path

1. Simple Path
2. Cycle Path

(i) Simple Path

Simple path is a path in which first and last vertex are different ($V0 \neq Vn$)

(ii) Cycle Path

Cycle path is a path in which first and last vertex are same (V0 = Vn).It is also called as Closed path.

Connected Graph

A graph G is said to be connected if there is a path between any two of its nodes.

Complete Graph

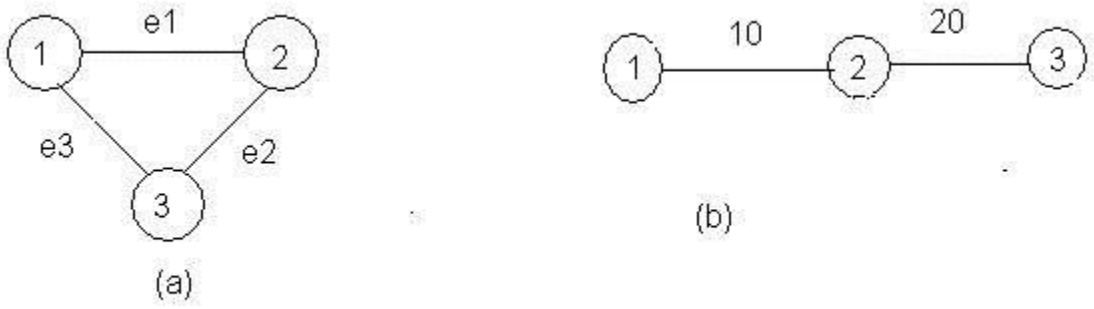
A graph G is said to be complete if every node u in G is adjacent to every other node v in G.

Tree

A connected graph T without any cycles is called a tree graph or free tree or, simply, a tree.

Labeled or Weighted Graph

If the weight is assigned to each edge of the graph then it is called as Weighted or Labeled graph.

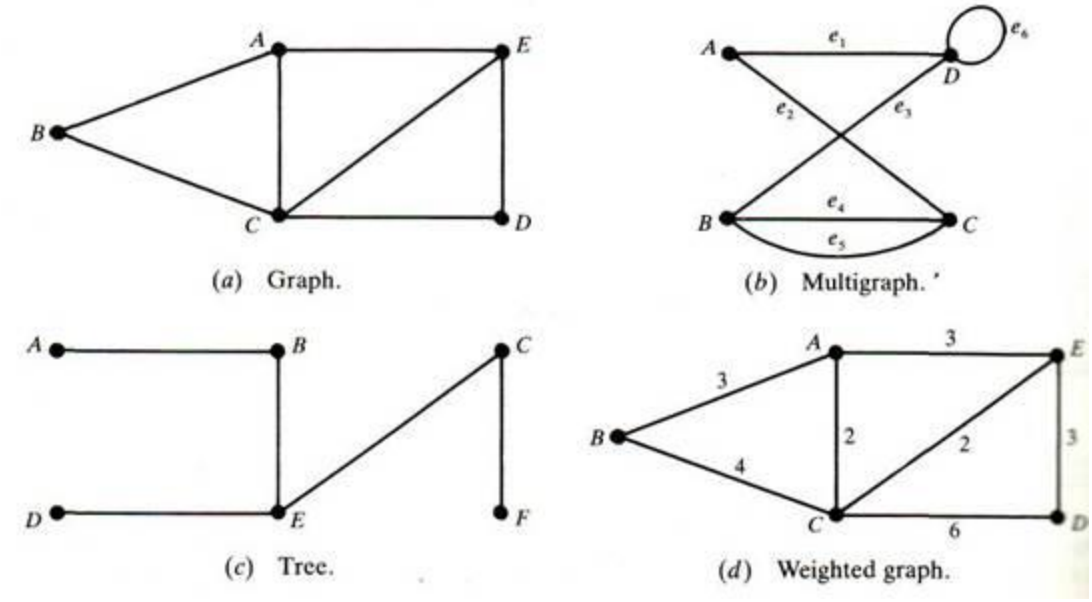


weighted or Labeled Graph

The definition of a graph may be generalized by permitting the following:

1. ***Multiple edges***: Distinct edges e and e' are called multiple edges if they connect the same endpoints, that is, if e = [u, v] and e' = [u, v].
2. ***Loops*:** An edge e is called a loop if it has identical endpoints, that is, if e = [u, u].

3. ***Finite Graph***:A multigraph M is said to be finite if it has a finite number of nodes and a finite number of edges.



Directed Graphs

A directed graph G, also called a digraph or graph is the same as a multigraph except that each edge e in G is assigned a direction, or in other words, each edge e is identified with an ordered pair (u, v) of nodes in G.

Outdegree and Indegree

Indegree : The indegree of a vertex is the number of edges for which v is head

Example

Indegree of 1 = 1

Indegree pf 2 = 2

Outdegree :The outdegree of a node or vertex is the number of edges for which v is tail.

Example

Outdegree of 1 =1

Outdegree of 2 =2

Simple Directed Graph

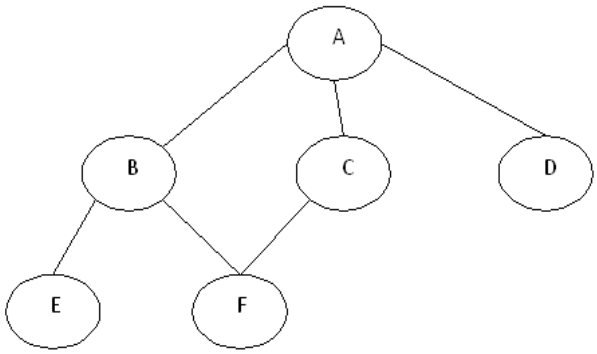
A directed graph G is said to be simple if G has no parallel edges. A simple graph G may have loops, but it cannot have more than one loop at a given node.

Graph Traversal

The breadth first search (BFS) and the depth first search (DFS) are the two algorithms used for traversing and searching a node in a graph. They can also be used to find out whether a node is reachable from a given node or not.

Depth First Search (DFS)

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search. Let's see how depth first search works with respect to the following graph:



As stated before, in DFS, nodes are visited by going through the depth of the tree from the starting node. If we do the depth first traversal of the above graph and print the visited node, it

will be “A B E F C D”. DFS visits the root node and then its children nodes until it reaches the end node, i.e. E and F nodes, then moves up to the parent nodes.

Algorithmic Steps

1. **Step 1**: Push the root node in the Stack.
2. **Step 2**: Loop until stack is empty.
3. **Step 3**: Peek the node of the stack.
4. **Step 4**: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack.
5. **Step 5**: If the node does not have any unvisited child nodes, pop the node from the stack.

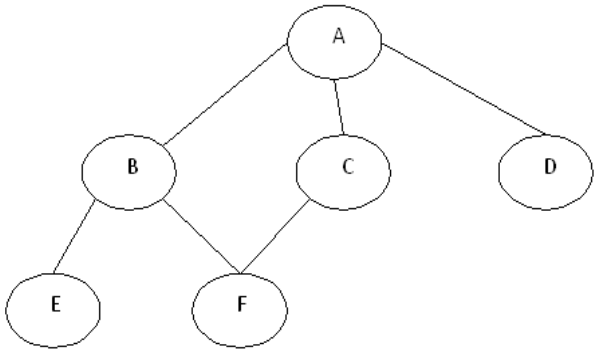
Based upon the above steps, the following Java code shows the implementation of the DFS algorithm:

```
public void dfs()
{
  //DFS uses Stack data structure
  Stack s=new Stack();
  s.push(this.rootNode);
  rootNode.visited=true;
  printNode(rootNode);
  while(!s.isEmpty())
  {
        Node n=(Node)s.peek();
        Node child=getUnvisitedChildNode(n);
        if(child!=null)
        {
              child.visited=true;
              printNode(child);
              s.push(child);
        }
```

```
            else
            {
                    s.pop();
            }
    }
    //Clear visited property of nodes
    clearNodes();
}
```

Breadth First Search (BFS)

This is a very different approach for traversing the graph nodes. The aim of BFS algorithm is to traverse the graph as close as possible to the root node. Queue is used in the implementation of the breadth first search. Let's see how BFS traversal works with respect to the following graph:



If we do the breadth first traversal of the above graph and print the visited node as the output, it will print the following output. "A B C D E F". The BFS visits the nodes level by level, so it will start with level 0 which is the root node, and then it moves to the next levels which are B, C and D, then the last levels which are E and F.

Algorithmic Steps

1. **Step 1**: Push the root node in the Queue.

2. **Step 2**: Loop until the queue is empty.
3. **Step 3**: Remove the node from the Queue.
4. **Step 4**: If the removed node has unvisited child nodes, mark them as visited and insert the unvisited children in the queue.

Based upon the above steps, the following Java code shows the implementation of the BFS algorithm:

```
public void bfs()
{
  //BFS uses Queue data structure
  Queue q=new LinkedList();
  q.add(this.rootNode);
  printNode(this.rootNode);
  rootNode.visited=true;
  while(!q.isEmpty())
  {
        Node n=(Node)q.remove();
        Node child=null;
        while((child=getUnvisitedChildNode(n))!=null)
        {
              child.visited=true;
              printNode(child);
              q.add(child);
        }
  }
  //Clear visited property of nodes
  clearNodes();
}
```

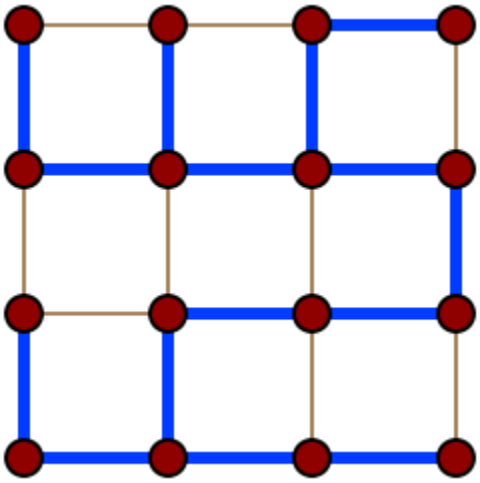
Spanning Trees:

In the mathematical field of graph theory, a **spanning tree** *T* of a connected, undirected graph *G* is a tree composed of all the vertices and some (or perhaps all) of the edges of *G*. Informally, a

spanning tree of *G* is a selection of edges of *G* that form a tree *spanning* every vertex. That is, every vertex lies in the tree, but no cycles (or loops) are formed. On the other hand, every bridge of *G* must belong to *T*.

A spanning tree of a connected graph *G* can also be defined as a maximal set of edges of *G* that contains no cycle, or as a minimal set of edges that connect all vertices.

Example:



A spanning tree (blue heavy edges) of a grid graph.

Spanning forests

A **spanning forest** is a type of subgraph that generalises the concept of a spanning tree. However, there are two definitions in common use. One is that a spanning forest is a subgraph that consists of a spanning tree in each connected component of a graph. (Equivalently, it is a maximal cycle-free subgraph.) This definition is common in computer science and optimisation. It is also the definition used when discussing minimum spanning forests, the generalization to disconnected graphs of minimum spanning trees. Another definition, common in graph theory, is that a spanning forest is any subgraph that is both a forest (contains no cycles) and spanning (includes every vertex).

Counting spanning trees

The number *t(G)* of spanning trees of a connected graph is an important invariant. In some cases, it is easy to calculate *t(G)* directly. It is also widely used in data structures in different computer

languages. For example, if G is itself a tree, then $t(G)=1$, while if G is the cycle graph C_n with n vertices, then $t(G)=n$. For any graph G, the number $t(G)$ can be calculated using Kirchhoff's matrix-tree theorem (follow the link for an explicit example using the theorem).

Cayley's formula is a formula for the number of spanning trees in the complete graph K_n with n vertices. The formula states that $t(K_n) = n^{n-2}$. Another way of stating Cayley's formula is that there are exactly n^{n-2} labelled trees with n vertices. Cayley's formula can be proved using Kirchhoff's matrix-tree theorem or via the Prüfer code.

If G is the complete bipartite graph $K_{p,q}$, then $t(G) = p^{q-1}q^{p-1}$, while if G is the n-dimensional hypercube graph Q_n, then $t(G) = 2^{2^n-n-1}\prod_{k=2}^{n} k^{\binom{n}{k}}$.

These formulae are also consequences of the matrix-tree theorem.

If G is a multigraph and e is an edge of G, then the number $t(G)$ of spanning trees of G satisfies the *deletion-contraction recurrence* $t(G)=t(G-e)+t(G/e)$, where $G-e$ is the multigraph obtained by deleting e and G/e is the contraction of G by e, where multiple edges arising from this contraction are not deleted.

Uniform spanning trees

A spanning tree chosen randomly from among all the spanning trees with equal probability is called a uniform spanning tree (UST). This model has been extensively researched in probability and mathematical physics.

Algorithms

The classic spanning tree algorithm, depth-first search (DFS), is due to Robert Tarjan. Another important algorithm is based on breadth-first search (BFS).

Planar Graphs:

In graph theory, a **planar graph** is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

A planar graph already drawn in the plane without edge intersections is called a **plane graph** or **planar embedding of the graph**. A plane graph can be defined as a planar graph with a mapping from every node to a point in 2D space, and from every edge to a plane curve, such that the extreme points of each curve are the points mapped from its end nodes, and all curves are disjoint except on their extreme points. Plane graphs can be encoded by combinatorial maps.

It is easily seen that a graph that can be drawn on the plane can be drawn on the sphere as well, and vice versa.

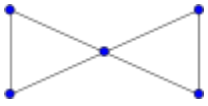
The equivalence class of topologically equivalent drawings on the sphere is called a **planar map**. Although a plane graph has an **external** or **unbounded** face, none of the faces of a planar map have a particular status.

Applications

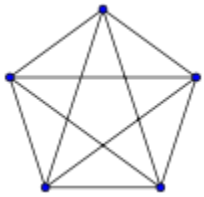
- Telecommunications – e.g. spanning trees
- Vehicle routing – e.g. planning routes on roads without underpasses
- VLSI – e.g. laying out circuits on computer chip.
- The puzzle game Planarity requires the player to "untangle" a planar graph so that none of its edges intersect.

Example graphs

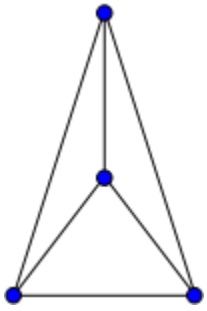
Planar **Nonplanar**

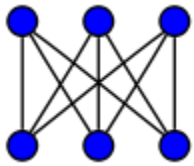


Butterfly graph



K_5





$K_{3,3}$

The complete graph

K_4 is planar

UNIT-VI

Graph Theory and Applications

Graph Theory and Applications:

Graphs are among the most ubiquitous models of both natural and human-made structures. They can be used to model many types of relations and process dynamics in physical, biological and social systems. Many problems of practical interest can be represented by graphs.

In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. One practical example: The link structure of a website could be represented by a directed graph. The vertices are the web pages available at the website and a directed edge from page *A* to page *B* exists if and only if *A* contains a link to *B*. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science. There, the transformation of graphs is often formalized and represented by graph rewrite systems. They are either directly used or properties of the rewrite systems (e.g. confluence) are studied. Complementary to graph transformation systems focussing on rule-based in-memory manipulation of graphs are graph databases geared towards transaction-safe, persistent storing and querying of graph-structured data.

Graph-theoretic methods, in various forms, have proven particularly useful in linguistics, since natural language often lends itself well to discrete structure. Traditionally, syntax and compositional semantics follow tree-based structures, whose expressive power lies in the Principle of Compositionality, modeled in a hierarchical graph. Within lexical semantics, especially as applied to computers, modeling word meaning is easier when a given word is understood in terms of related words; semantic networks are therefore important in computational linguistics. Still other methods in phonology (e.g. Optimality Theory, which uses lattice graphs) and morphology (e.g. finite-state morphology, using finite-state transducers) are common in the analysis of language as a graph. Indeed, the usefulness of this area of mathematics to linguistics has borne organizations such as TextGraphs, as well as various 'Net' projects, such as WordNet, VerbNet, and others.

Graph theory is also used to study molecules in chemistry and physics. In condensed matter physics, the three dimensional structure of complicated simulated atomic structures can be studied quantitatively by gathering statistics on graph-theoretic properties related to the topology of the atoms. For example, Franzblau's shortest-path (SP) rings. In chemistry a graph makes a natural model for a molecule, where vertices represent atoms and edges bonds. This approach is especially used in computer processing of molecular structures, ranging from chemical editors to database searching. In statistical physics, graphs can represent local connections between interacting parts of a system, as well as the dynamics of a physical process on such systems.

Graph theory is also widely used in sociology as a way, for example, to measure actors' prestige or to explore diffusion mechanisms, notably through the use of social network analysis software.Likewise, graph theory is useful in biology and conservation efforts where a vertex can represent regions where certain species exist (or habitats) and the edges represent migration paths, or movement between the regions. This information is important when looking at breeding patterns or tracking the spread of disease, parasites or how changes to the movement can affect other species.

In mathematics, graphs are useful in geometry and certain parts of topology, e.g. Knot Theory. Algebraic graph theory has close links with group theory.

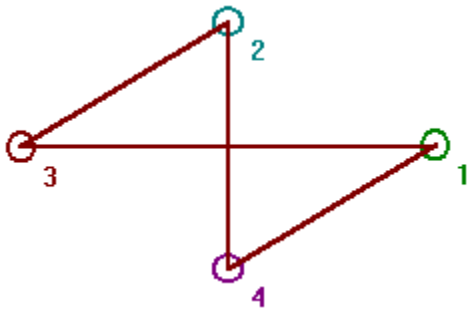
A graph structure can be extended by assigning a weight to each edge of the graph. Graphs with weights, or weighted graphs, are used to represent structures in which pairwise connections have some numerical values. For example if a graph represents a road network, the weights could represent the length of each road.

Basic Concepts Isomorphism:

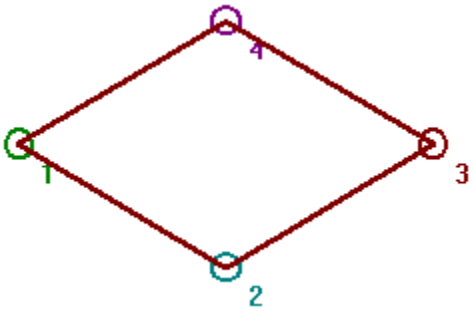
Let G1 and G1 be two graphs and let f be a function from the vertex set of G1 to the vertex set of G2. Suppose that f is one-to-one and onto & f(v) is adjacent to f(w) in G2 if and only if v is adjacent to w in G1.

Then we say that the function f is an isomorphism and that the two graphs G1 and G2 are isomorphic. So two graphs G1 and G2 are isomorphic if there is a one-to-one correspondence between vertices of G1 and those of G2 with the property that if two vertices of G1 are adjacent

then so are their images in G2. If two graphs are isomorphic then as far as we are concerned they are the same graph though the location of the vertices may be different. To show you how the program can be used to explore isomorphism draw the graph in figure 4 with the program (first get the null graph on four vertices and then use the right mouse to add edges).



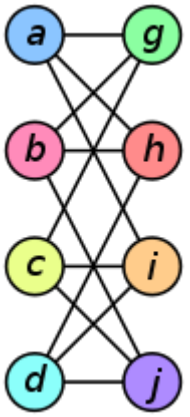
Save this graph as Graph 1 (you need to click Graph then Save). Now get the circuit graph with 4 vertices. It looks like figure 5, and we shall call it C(4).

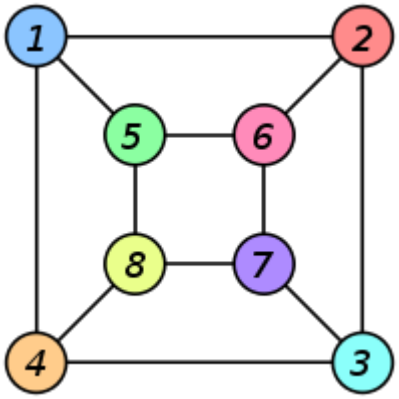


Example:

The two graphs shown below are isomorphic, despite their different looking drawings.

Graph G	**Graph H**	**An isomorphism between G and H**





$f(a) = 1$

$f(b) = 6$

$f(c) = 8$

$f(d) = 3$

$f(g) = 5$

$f(h) = 2$

$f(i) = 4$

$f(j) = 7$

Subgraphs:

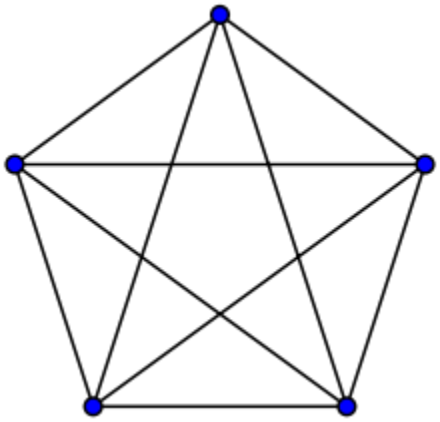
A **subgraph** of a graph *G* is a graph whose vertex set is a subset of that of *G*, and whose adjacency relation is a subset of that of *G* restricted to this subset. In the other direction, a **supergraph** of a graph *G* is a graph of which *G* is a subgraph. We say a graph *G* **contains** another graph *H* if some subgraph of *G* is *H* or is isomorphic to *H*.

A subgraph *H* is a **spanning subgraph**, or **factor**, of a graph *G* if it has the same vertex set as *G*. We say *H* spans *G*.

A subgraph *H* of a graph *G* is said to be **induced** if, for any pair of vertices *x* and *y* of H, *xy* is an edge of *H* if and only if *xy* is an edge of G. In other words, *H* is an induced subgraph of *G* if it has all the edges that appear in *G* over the same vertex set. If the vertex set of *H* is the subset *S* of *V(G)*, then *H* can be written as *G*[*S*] and is said to be **induced by *S***.

A graph that does *not* contain *H* as an induced subgraph is said to be ***H*-free**.

A **universal graph** in a class ***K*** of graphs is a simple graph in which every element in ***K*** can be embedded as a subgraph.



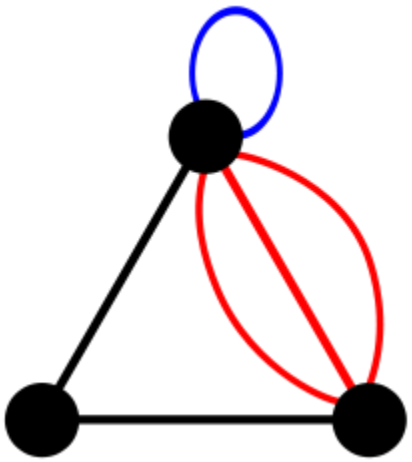
K_5, a complete graph. If a subgraph looks like this, the vertices in that subgraph form a clique of size 5.

Multi graphs:

In mathematics, a **multigraph** or **pseudograph** is a graph which is permitted to have multiple edges, (also called "parallel edges"), that is, edges that have the same end nodes. Thus two vertices may be connected by more than one edge. Formally, a multigraph G is an ordered pair $G:=(V, E)$ with

- V a set of *vertices* or *nodes*,
- E a multiset of unordered pairs of vertices, called *edges* or *lines*.

Multigraphs might be used to model the possible flight connections offered by an airline. In this case the multigraph would be a directed graph with pairs of directed parallel edges connecting cities to show that it is possible to fly both *to* and *from* these locations.



A multigraph with multiple edges (red) and a loop (blue). Not all authors allow multigraphs to have loops.

Euler circuits:

In graph theory, an **Eulerian trail** is a trail in a graph which visits every edge exactly once. Similarly, an **Eulerian circuit** is an Eulerian trail which starts and ends on the same vertex. They were first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736. Mathematically the problem can be stated like this:

Given the graph on the right, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) which visits each edge exactly once?

Euler proved that a necessary condition for the existence of Eulerian circuits is that all vertices in the graph have an even degree, and stated without proof that connected graphs with all vertices of even degree have an Eulerian circuit. The first complete proof of this latter claim was published in 1873 by Carl Hierholzer.

The term **Eulerian graph** has two common meanings in graph theory. One meaning is a graph with an Eulerian circuit, and the other is a graph with every vertex of even degree. These definitions coincide for connected graphs.

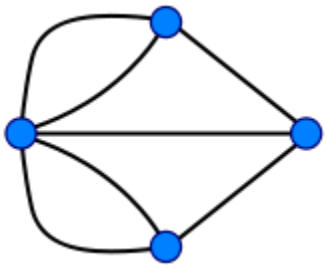
For the existence of Eulerian trails it is necessary that no more than two vertices have an odd degree; this means the Königsberg graph is *not* Eulerian. If there are no vertices of odd degree, all Eulerian trails are circuits. If there are exactly two vertices of odd degree, all Eulerian trails start at one of them and end at the other. Sometimes a graph that has an Eulerian trail but not an Eulerian circuit is called **semi-Eulerian**.

An **Eulerian trail**, **Eulerian trail** or **Euler walk** in an undirected graph is a path that uses each edge exactly once. If such a path exists, the graph is called **traversable** or **semi-eulerian**.

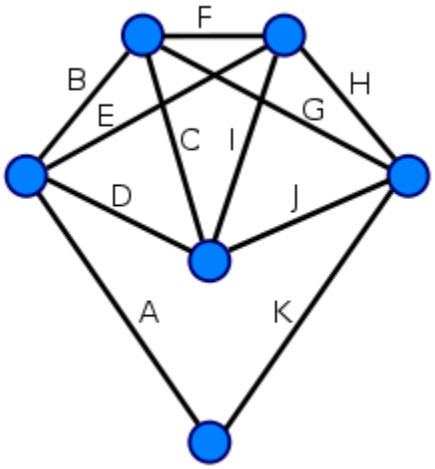
An **Eulerian cycle**, **Eulerian circuit** or **Euler tour** in an undirected graph is a cycle that uses each edge exactly once. If such a cycle exists, the graph is called **unicursal**. While such graphs are Eulerian graphs, not every Eulerian graph possesses an Eulerian cycle.

For directed graphs path has to be replaced with directed path and cycle with directed cycle.

The definition and properties of Eulerian trails, cycles and graphs are valid for multigraphs as well.



This graph is not Eulerian, therefore, a solution does not exist.



Every vertex of this graph has an even degree, therefore this is an Eulerian graph. Following the edges in alphabetical order gives an Eulerian circuit/cycle.

Hamiltonian graphs:

In the mathematical field of graph theory, a **Hamiltonian path** (or **traceable path**) is a path in an undirected graph which visits each vertex exactly once. A **Hamiltonian cycle** (or **Hamiltonian circuit**) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem which is NP-complete.

Hamiltonian paths and cycles are named after William Rowan Hamilton who invented the Icosian game, now also known as *Hamilton's puzzle*, which involves finding a Hamiltonian cycle in the edge graph of the dodecahedron. Hamilton solved this problem using the Icosian Calculus, an algebraic structure based on roots of unity with many similarities to the quaternions (also invented by Hamilton). This solution does not generalize to arbitrary graphs.

A *Hamiltonian path* or *traceable path* is a path that visits each vertex exactly once. A graph that contains a Hamiltonian path is called a **traceable graph**. A graph is **Hamilton-connected** if for every pair of vertices there is a Hamiltonian path between the two vertices.

A *Hamiltonian cycle*, *Hamiltonian circuit*, *vertex tour* or *graph cycle* is a cycle that visits each vertex exactly once (except the vertex which is both the start and end, and so is visited twice). A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**.

Similar notions may be defined for *directed graphs*, where each edge (arc) of a path or cycle can only be traced in a single direction (i.e., the vertices are connected with arrows and the edges traced "tail-to-head").

A **Hamiltonian decomposition** is an edge decomposition of a graph into Hamiltonian circuits.

Examples

- a complete graph with more than two vertices is Hamiltonian
- every cycle graph is Hamiltonian
- every tournament has an odd number of Hamiltonian paths
- every platonic solid, considered as a graph, is Hamiltonian

Chromatic Numbers:

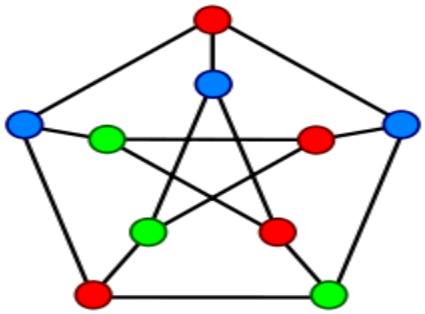
In graph theory, **graph coloring** is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a **vertex coloring**. Similarly, an **edge coloring** assigns a color to each edge so that no two adjacent edges share the same color, and a **face coloring** of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a planar graph is just a vertex coloring of its planar dual. However, non-vertex coloring problems are often stated and studied *as is*. That is partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a graph embedded in the plane. By planar duality it became coloring the vertices, and in this form it generalizes to all graphs. In mathematical and computer representations it is typical to use the first few positive or

nonnegative integers as the "colors". In general one can use any finite set as the "color set". The nature of the coloring problem depends on the number of colors but not on what they are.

Graph coloring enjoys many practical applications as well as theoretical challenges. Beside the classical types of problems, different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself. It has even reached popularity with the general public in the form of the popular number puzzle Sudoku. Graph coloring is still a very active field of research.



A proper vertex coloring of the Petersen graph with 3 colors, the minimum number possible.

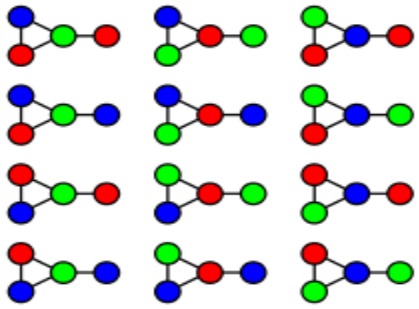
Vertex coloring

When used without any qualification, a **coloring** of a graph is almost always a *proper vertex coloring*, namely a labelling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color. Since a vertex with a loop could never be properly colored, it is understood that graphs in this context are loopless.

The terminology of using *colors* for vertex labels goes back to map coloring. Labels like *red* and *blue* are only used when the number of colors is small, and normally it is understood that the labels are drawn from the integers {1,2,3,...}.

A coloring using at most k colors is called a (proper) ***k*-coloring**. The smallest number of colors needed to color a graph G is called its **chromatic number**, $\chi(G)$. A graph that can be assigned a (proper) k-coloring is ***k*-colorable**, and it is ***k*-chromatic** if its chromatic number is exactly k. A subset of vertices assigned to the same color is called a *color class*, every such class forms an

independent set. Thus, a *k*-coloring is the same as a partition of the vertex set into *k* independent sets, and the terms *k-partite* and *k-colorable* have the same meaning.



This graph can be 3-colored in 12 different ways.

The following table gives the chromatic number for familiar classes of graphs.

graph G	$\gamma(G)$
complete graph K_n	n
cycle graph C_n, $n > 1$	$\begin{cases} 3 & \text{for n odd} \\ 2 & \text{for n even} \end{cases}$
star graph S_n, $n > 1$	2
wheel graph W_n, $n > 2$	$\begin{cases} 3 & \text{for n odd} \\ 4 & \text{for n even} \end{cases}$