RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL
New Scheme Based On AICTE Flexible Curricula
Information Technology, III-Semester
IT 301 Discrete Structure

Course objectives

The main objectives of this course are:

1. To introduce students with sets, relations, functions, graph, and probability.
2. To enable students to perform set operation and solve logical reasoning and verify the correctness of logical statement.
3. To apply the properties of relations and find partially ordered set and lattices.

Unit I-Set Theory, Relation, Function, Theorem Proving Techniques : Set Theory: Definition of sets, countable and uncountable sets, Venn Diagrams, proofs of some general identities on sets Relation: Definition, types of relation, composition of relations, Pictorial representation of relation, Equivalence relation, Partial ordering relation, Job Scheduling problem

Function: Definition, type of functions, one to one, into and onto function, inverse function, composition of functions, recursively defined functions, pigeonhole principle. Theorem proving Techniques: Mathematical induction, Proof by contradiction.

Unit II- Algebraic Structures: Definition, Properties, types: Semi Groups, Monoid, Groups, Abelian group, properties of groups, Subgroup, cyclic groups, Normal subgroup, Homomorphism and isomorphism of Groups, example and standard results, Rings and Fields: definition and standard results.

Unit III- Propositional Logic: Proposition, First order logic, Basic logical operation, truth tables, tautologies, Contradictions, Algebra of Proposition, logical implications, logical equivalence, predicates, Normal Forms, Universal and existential quantifiers. Introduction to finite state machine Finite state machines as models of physical system equivalence machines, Finite state machines as language recognizers

Unit IV- Graph Theory: Introduction and basic terminology of graphs, Planer graphs, Multigraphs and weighted graphs, Isomorphic graphs, Paths, Cycles and connectivity, Shortest path in weighted graph, Introduction to Eulerian paths and circuits, Hamiltonian paths and circuits, Graph coloring, chromatic number, Isomorphism and Homomorphism of graphs.

Unit V- Posets, Hasse Diagram and Lattices: Introduction, ordered set, Hasse diagram of partially, ordered set, isomorphic ordered set, well ordered set, properties of Lattices, bounded and complemented lattices. Combinatorics: Introduction, Permutation and combination, Binomial Theorem, Recurrence Relation and Generating Function: Introduction to Recurrence Relation and Recursive algorithms , Linear recurrence relations with constant coefficients, Homogeneous solutions, Particular solutions, Total solutions , Generating functions , Solution by method of generating functions.

Reference Books:

1. C.L.Liu" Elements of Discrere Mathematics" TMH.
2. Lipschutz, "Discrete mathematics (Schaum)",TMH.
3. U.S Gupta " Discrete Mathematical Structures" Pearson.
4. S. Santha," Discrete Mathematics with Combinatorics and graph theory", Cengage Learning.
5. Dr.Sukhendu. Dey " Graph Theory With Applications" Shroff Publishers

Course Outcomes

On completion of the course;
1. Students will be able to understand the notion of mathematical thinking, and algorithmic thinking and be able to apply them in problem solving such as formal specification, verification, and basic concepts of set theory.
2. Students understand the basic principle of Boolean algebra, logic and set theory.
3. Be able to construct simple mathematical proof and possess the ability to verify them.

UNIT-1

Set - Definition

A set is an unordered collection of different elements. A set can be written explicitly by listing its elements using set bracket. If the order of the elements is changed or any element of a set is repeated, it does not make any changes in the set.

Some Example of Sets

1. A set of all positive integers
2. A set of all the planets in the solar system
3. A set of all the states in India
4. A set of all the lowercase letters of the alphabet

Representation of a Set

Sets can be represented in two ways −

1. ʻoster or Tabular Form
2. Set Builder Notation

1. Roster or Tabular Form

The set is represented by listing all the elements comprising it. The elements are enclosed within braces and separated by commas.

Example 1 − Set of vowels in English alphabet, *A*= {*a*, *e*, *i*, *o*, *u*}

Example 2 − Set of odd numbers less than 10, *B*= {1, 3, 5, 7, 9}

2. Set Builder Notation

The set is defined by specifying a property that elements of the set have in common. The set is described as ***A*={*x: p*(*x*)}**

Example 1 − the set {*a*, *e*, *i*, *o*, *u*} is written as −

A={*x*: x is a vowel in English alphabet}

Example 2 − the set {1, 3, 5, 7, 9} is written as −

B={*x*: 1≤*x*<10 *and* (*x*%2) ≠ 0}

- If an element x is a member of any set S, it is denoted by *x*∈ *S*
- If an element y is not a member of set S, it is denoted by *y*ø *S*.

Example 3− If *S*= {1, 1.2, 1.7, 2}, 1∈*S b*ut 1.5ø*S*

Some Important Sets

N − the set of all natural numbers = {1, 2, 3, 4 …}

) − the set of all integers = {....., −3, −2, −1, 0, 1, 2, 3.....}

)$^{+}$ − the set of all positive integers

Q − the set of all rational numbers

R − the set of all real numbers

W − the set of all whole numbers

Cardinality of a Set

Cardinality of a set S, denoted by |*S*|, is the number of elements of the set. The number is also referred as the cardinal number. If a set has an infinite number of elements, its cardinality is ∞.

Example − |{1,4,3,5}|=4,|{1,2,3,4,5,…}|=∞

If there are two sets X and Y,

1. |X|=|Y| denotes two sets X and Y having same cardinality. It occurs when the number of elements in X is exactly equal to the number of elements in Y. In this case, there exists a bijective function ‘f’ from X to Y.
2. X|≤|Y denotes that set X’s cardinality is less than or equal to set Y’s cardinality. It occurs when number of elements in X is less than or equal to that of Y. Here, there exists an injective function ‘f’ from X to Y.

3. |X|<|Y| denotes that set X's cardinality is less than set Y's cardinality. It occurs when number of elements in X is less than that of Y. Here, the function 'f' from X to Y is injective function but not bijective.
4. If |X|≤|Y| and |X|≤|Y| then |X|=|Y| the sets X and Y are commonly referred as equivalent sets.

Types of Sets

Sets can be classified into many types. Some of which are finite, infinite, subset, universal, proper, singleton set, etc.

1. Finite Set- A set which contains a definite number of elements is called a finite set.

Example – $S = \{x \mid x \in N \text{ and } 70>x>50\}$

2. Infinite Set- A set which contains infinite number of elements is called an infinite set.

Example – $S=\{x \mid x \in N \text{ and } x>10\}$

3. Countable set: a set A is called countable infinite if A ∈ N. We say that A is countable if A∈N or A is finite. A countable set is a set with the same cardinality (number of elements) as some subset of the set of natural numbers. A countable set is either a finite set or a countable infinite set. Whether finite or infinite, the elements of a countable set can always be counted one at a time and, although the counting may never finish, every element of the set is associated with a unique natural number.

4. Uncountable Sets: An uncountable set (or unaccountably infinite set) is an infinite set that contains too many elements to be countable. The unaccountability of a set is closely related to its cardinal number: a set is uncountable if its cardinal number is larger than that of the set of all natural numbers.

5. **Subset-** A set X is a subset of set Y (Written as $X \subseteq Y$) if every element of X is an element of set Y.

Example 1 – Let, $X=\{1, 2, 3, 4, 5, 6\}$ and $Y=\{1, 2\}$. Here set Y is a subset of set X as all the elements of set Y is in set X. Hence, we can write $Y \subseteq X$

Example 2 – Let, $X= \{1, 2, 3\}$ and $Y= \{1, 2, 3\}$. Here set Y is a subset (Not a proper subset) of set X as all the elements of set Y is in set X. Hence, we can write $Y \subseteq X$

6. Proper Subset- The term "proper subset" can be defined as "subset of but not equal to". A Set X is a proper subset of set Y (Written as $X \subset Y$) if every element of X is an element of set Y and $|X|<|Y|$

Example – Let, $X=\{1,2,3,4,5,6\}$ and $Y=\{1,2\}$. Here set $Y \subset X$ since all elements in Y are contained in X too and X has at least one element is more than set Y

7. Universal Set-It is a collection of all elements in a particular context or application. All the sets in that context or application are essentially subsets of this universal set. Universal sets are represented as U

Example – we may define U as the set of all animals on earth. In this case, set of all mammals is a subset of U, set of all fishes is a subset of U, and set of all insects is a subset of U, and so on.

8. Empty Set or Null Set- An empty set contains no elements. It is denoted by ∅ . As the number of elements in an empty set is finite, empty set is a finite set. The cardinality of empty set or null set is zero.

Example – $S= \{x \mid x \in N \text{ and } 7<x<8\} =\emptyset$

9. Singleton Set or Unit Set- Singleton set or unit set contains only one element. A singleton set is denoted by $\{s\}$

Example – $S= \{x \mid x \in N, 7<x<9\} = \{8\}$

10. Equal Set- If two sets contain the same elements they are said to be equal.

Example – If $A=\{1,2,6\}$ and $B=\{6,1,2\}$ they are equal as every element of set A is an element of set B and every element of set B is an element of set A.

11. Equivalent Set- If the cardinalities of two sets are same, they are called equivalent sets.

Example – If $A= \{1, 2, 6\}$ and $B=\{16,17,22\}$, they are equivalent as cardinality of A is equal to the cardinality of B. i.e. $|A|=|B|=3$

12. Overlapping Set - Two sets that have at least one common element are called overlapping sets.

In case of overlapping sets –

a) $n(A \cup B)=n(A)+n(B)-n(A \cap B)$
b) $n(A \cup B)=n(A-B)+n(B-A)+n(A \cap B)$
c) $n(A)=n(A-B)+n(A \cap B)$
d) $n(B)=n(B-A)+n(A \cap B)$

Example – Let, $A= \{1, 2, 6\}$ and $B= \{6, 12, 42\}$

There is a common element '6'; hence these sets are overlapping sets.

13. Disjoint Set - Two sets A and B are called disjoint sets if they do not have even one element in common. Therefore, disjoint sets have the following properties −

n (A∩B)=∅

n (AUB)=n(A)+n(B)

Example − Let, *A*= {1, 2, 6} and *B*= {7, 9, 14} there is not a single common element, hence these sets are overlapping sets.

Venn Diagrams

Set Operations

Set Operations include Set Union, Set Intersection, Set Difference, Complement of Set, and Cartesian product.

1. Set Union - The union of sets A and B (denoted by *AUB)* is the set of elements which are in A, in B, or in both A and B. Hence, *AUB*={*x* |*x* ∈*A OR x* ∈*B*}

Example − If *A*= {10, 11, 12, 13} and B = {13, 14, 15}, then *AUB*= {10, 11, 12, 13, 14, 15}

(The common element occurs only once)

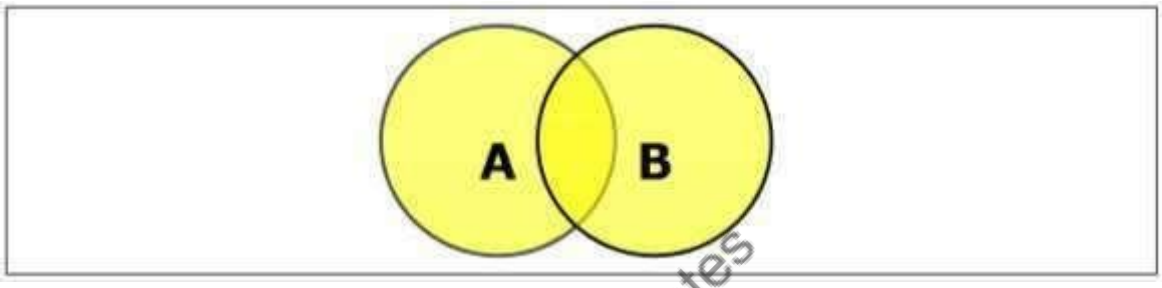


Fig 1.1 Union

2. Set Intersection - The intersection of sets A and B (denoted by *A∩B)* is the set of elements which are in both A and B. Hence, *A∩B*={*x* |*x* ∈*A AND x* ∈*B*}

Example − If *A*= {11, 12, 13} and *B* = {13,14,15}, then *A∩B*={13}

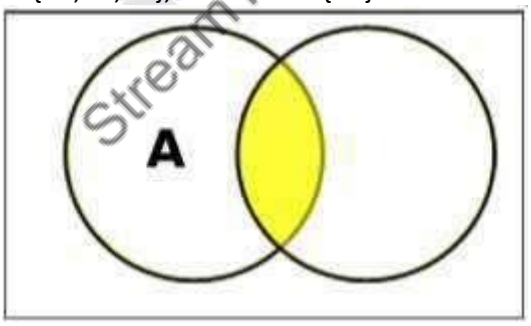


Fig 1.2 Intersection

3. Set Difference/ Relative Complement - The set difference of sets A and B (denoted by *A–B*) is the set of elements which are only in A but not in B. Hence, *A−B*={*x* |*x* ∈*A AND x* ø*B*}

Example − If *A*= {10, 11, 12, 13}

and *B*={13,14,15}, then (*A−B*)={10,11,12} and (*B−A*)={14,15}. Here, we can see (*A−B*) ≠ (*B−A*)

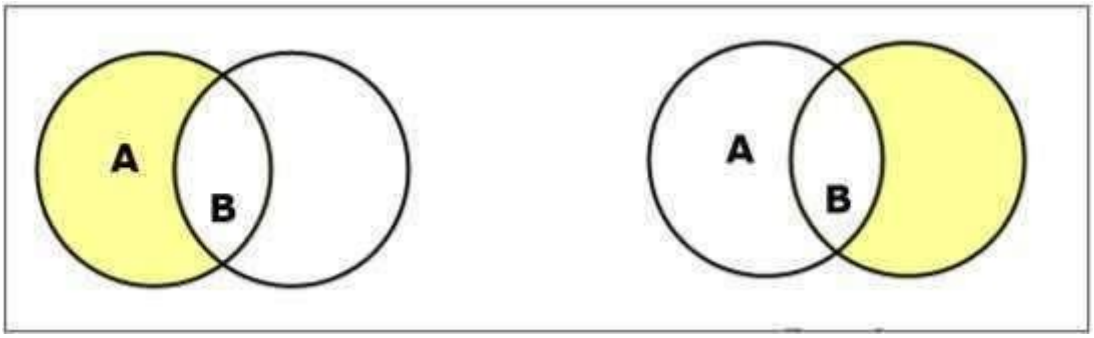


Fig 1.3 Set Differences

4. Complement of a Set - The complement of a set A (denoted by *A*') is the set of elements which are not in set A. Hence, *A*'={*x* | *x* ø*A*}

More specifically, $A'= (U-A)$ where U is a universal set which contains all objects.

Example − If $A= \{x \mid x \text{ belongs to set of odd integers}\}$ then $A'=\{y \mid y \text{ does not belong to set of odd integers}\}$

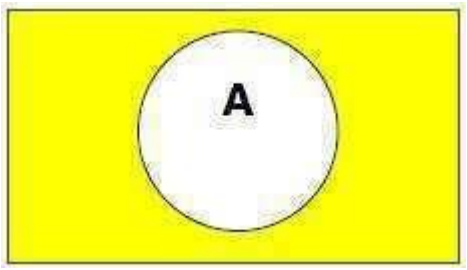


Fig 1.4 **Complement**

5. Cartesian Product / Cross Product - The Cartesian product of n number of sets $A1,A2,\ldots An$ denoted as $A1\times A2\cdots\times An$ can be defined as all possible ordered pairs $(x1,x2\ldots xn)$ where $x1\in A1,x2\in A2,\ldots xn\in An$

Example − If we take two sets $A= \{a, b\}$ and $B= \{1, 2\}$

The Cartesian product of A and B is written as − $A\times B= \{(a, 1), (a, 2), (b, 1), (b, 2)\}$

The Cartesian product of B and A is written as − $B\times A= \{(1, a), (1, b), (2, a), (2, b)\}$

6. Power Set - Power set of a set S is the set of all subsets of S including the empty set. The cardinality of a power set of a set S of cardinality n is 2^n. Power set is denoted as $P(S)$

Example –

For a set $S= \{a, b, c, d\}$

Let us calculate the subsets −

1. Subsets with 0 elements − $\{\emptyset\}$ (the empty set)
2. Subsets with 1 element − $\{a\},\{b\},\{c\},\{d\}$
3. Subsets with 2 elements − $\{a, b\},\{a, c\},\{a, d\},\{b, c\},\{b, d\},\{c, d\}$
4. Subsets with 3 elements − $\{a, b, c\},\{a, b, d\},\{a, c, d\},\{b, c, d\}$
5. Subsets with 4 elements − $\{a, b, c, d\}$

Hence, $P(S)=\{\{\emptyset\},\{a\},\{b\},\{c\},\{d\},\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{b,d\},\{c,d\},\{a,b,c\},\{a,b,d\},\{a,c,d\},\{b,c,d\},\{a,b,c,d\}\}$

$|P(S)|=2^4=16$

Note − the power set of an empty set is also an empty set. $|P(\{\emptyset\})|=2^0=1$

Relations

Whenever sets are being discussed, the relationship between the elements of the sets is the next thing that comes up. **Relations** may exist between objects of the same set or between objects of two or more sets.

Definition and Properties

A binary relation ' from set x to y (written as $x\ R\ y$ or $R(x, y)$) is a subset of the Cartesian product $x \times y$. If the ordered pair of G is reversed, the relation also changes. Generally an n-ary relation ' between sets $A1\ldots$ and An is a subset of the n-ary product $A1\times\cdots\times An$. The minimum cardinality of a relation ' is)ero and maximum is $n2$ in this case. A binary relation ' on a single set A is a subset of $A \times A$

For two distinct sets, A and B, having cardinalities m and n respectively, the maximum cardinality of a relation ' from A to B is mn.

Domain and Range

If there are two sets A and B, and relation ' have order pair (x, y), then −

1. The **domain** of ', Dom('), is the set {x|(x, y) ∈ ' for some y in B}
2. The **range** of ', 'an('), is the set {y|(x, y) ∈ ' for some x in A}

Examples Let, $A = \{1, 2, 9\}$ and $B = \{1, 3, 7\}$

Case 1 − If relation ' is 'equal to' then $R= \{(1, 1), (3, 3)\}$

Dom(') = {1, 3}, $Ran(R) = \{1, 3\}$

Case 2 − If relation ' is 'less than' then $R= \{(1, 3), (1, 7), (2, 3), (2, 7)\}$

Dom(') = {1, 2}, $Ran(R) = \{3, 7\}$

Case 3 − If relation ' is 'greater than' then $R= \{(2, 1), (9, 1), (9, 3), (9, 7)\}$

Dom(') = {2, 9}, $Ran(R) = \{1, 3, 7\}$

Types of Relations

1. The **Empty Relation** between sets X and Y, or on E, is the empty set ∅

2. The **Full Relation** between sets X and Y is the set $X \times Y$

3. The **Identity Relation** on set X is the set $\{(x, x) | x \in X\}$

4. The Inverse ‘elation ‘' of a relation ‘ is defined as − $R' = \{(b, a) | (a, b) \in R\}$

Example − If $R = \{(1, 2), (2, 3)\}$ then R' will be $\{(2, 1), (3, 2)\}$

Properties of Relation:-

1. A relation ‘ on set A is called **Reflexive** if $\forall a \in A$ is related to a (a‘a holds)

Example − The relation $R = \{(a, a),(b, b)\}$ on set $X = \{a, b\}$ is reflexive.

2. A relation ‘ on set A is called **Irreflexive** if no $a \in A$ is related to a (a‘a does not hold).

Example − The relation $R = \{(a, b),(b, a)\}$ on set $X = \{a, b\}$ is irreflexive.

3. A relation ‘ on set A is called **Symmetric** if xRy implies yRx, $\forall x \in A$ and $\forall y \in A$

Example − The relation $R = \{(1,2),(2,1),(3,2),(2,3)\}$ on set $A = \{1,2,3\}$ is symmetric.

4. A relation ‘ on set A is called **Anti-Symmetric** if xRy and yRx implies $x = y$ $\forall x \in A$ and $\forall y \in A$

Example − the relation $R = \{(x, y) \rightarrow N | x \leq y\}$ is anti-symmetric since $x \leq y$ and $y \leq x$ implies $x = y$

5. A relation ‘ on set A is called **Transitive** if xRy and yRz implies $xRz, \forall x, y, z \in A$

Example − The relation $R = \{(1,2),(2,3),(1,3)\}$ on set $A = \{1, 2, 3\}$ is transitive.

Composition of Relations: Suppose that we have three sets A, B and C. A relation ‘ defined from A to B, and a relation S defined from B to C. We can now define a new relation known as the composition of ‘ and S, written as S o ‘. This new relation is defined as follows:

If a is an element in A and c is an element in C, then a(S o ‘)c, if and only if, there exists some element b in B, such that a‘b and bSc. This means that, we have a relation S o ‘ from a to c, if and only if, we can reach from a to c in two steps; i.e. from a to b related by ‘ and from b to c related by S. In this manner, relation S o ‘ can be interpreted as ‘ followed by S, since this is the order in which the two relations need to be considered, first ‘ then S.

Question: Let A = {a, b, c}, B = {1, 2} and C = {a, b, g} is being three sets and R = {(a, 1), (a, 2), (b, 2), (c, 1)} and S = {(1, a), (2, b), (2, g)} be two relations, find SoR.

Solution:

Here, R is a relation where a goes to 1 and so on. Thus, S o R will be calculated as a goes to 1 and in turn, 1 goes to a, yielding (a, a).

Hence, SoR = {(a, a), (a, b), (a, g), (b, b), (b, g), (c, a)}.

Representation of Relations using Graph

A relation can be represented using a directed graph. The number of vertices in the graph is equal to the number of elements in the set from which the relation has been defined. For each ordered pair (x, y) in the relation ‘, there will be a directed edge from the vertex ‘x’ to vertex ‘y’. If there is an ordered pair (x, x), there will be self- loop on vertex ‘x’.

Suppose, there is a relation ‘= {(1, 1), (1, 2), (3, 2)} on set S = {1, 2, 3} , it can be represented by the following graph −

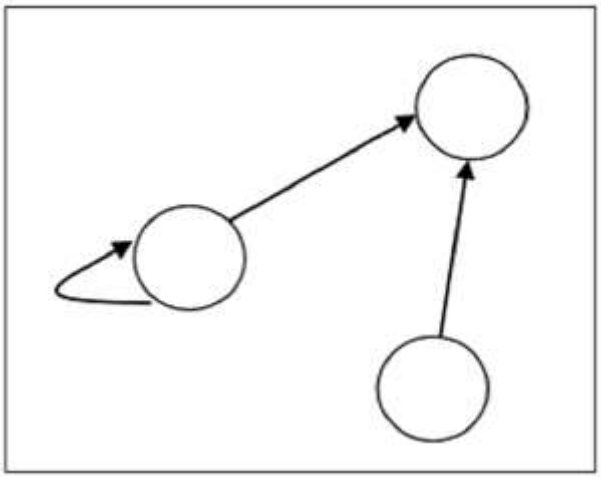


Fig 1.5

Equivalence Relation:
A relation is an Equivalence ʻelation if it is reflexive, symmetric, and transitive.
Example 1 − The relation *R*={(1,1),(2,2),(3,3),(1,2),(2,1),(2,3),(3,2),(1,3),(3,1)} on set *A*={1,2,3} is an equivalence relation since it is reflexive, symmetric, and transitive.
Example 2: The equality relation (=) on a set of numbers such as {1, 2, 3} is an equivalence relation.
Example 3: The congruent modulo m relation on the set of integers i.e. {<a, b>| a b (mod m)}, where m is a positive integer greater than 1, is an equivalence relation.

Partial Order Relation:
A relation is a Partial Order ʻelation if it is reflexive, Anti symmetric, and transitive.
Example 1 − The relation *R*={(1,1),(2,2),(3,3),(1,2),(2,1),(2,3),(3,2),(1,3),(3,1)} on set *A*={1,2,3} is an equivalence relation since it is reflexive, symmetric, and transitive.
Example 2 − A relation "$\leq$" is a partial order on a set S if it has:
1. ʻeflexivity: for all.
2. Anti-symmetric: and implies.
3. Transitivity: and implies.

Job Scheduling: Suppose we have n jobs each of which take time t_i to process and m identical machines on which to schedule their completion Jobs cannot be split between machines For a given scheduling, let A_j be the set of jobs assigned to machine j. Let $L_j = P_{i \in Aj}\, t_i$ is the load of machine j. The minimum make span scheduling problem is to find an assignment of jobs to machines that minimizes the make span, defined as the maximum load over all machines (i.e. maxj L_j).
We consider the following greedy algorithm for this problem which sorts the jobs so that t1 ≥ t2 ≥ ... ≥ tn, and iteratively allocates the next job to the machine with the least load.

Scheduling Problems: In order to make our problem more specific and for simplicity, we need to make some assumptions. First we will assume that any processor can work on any task. In addition we will always assume the following two rules:
(i) No processor can be idle if there is some task it can be doing.
(ii) Once a processor has begun a task, it alone must continue to process that task until it is finished.

Discrete Mathematics - Functions
A **Function** assigns to each element of a set, exactly one element of a related set. Functions find their application in various fields like representation of the computational complexity of algorithms, counting objects, study of sequences and strings, to name a few. The third and final chapter of this part highlights the important aspects of functions.
Function - Definition
A function or mapping (Defined as $f: X \to Y$) is a relationship from elements of one set X to elements of another set Y (X and Y are non-empty sets). X is called Domain and Y is called Codomain of function 'f'. Function 'f' is a relation on X and Y such that for each $x \in X$, there exists a unique $y \in Y$ such that $(x, y) \in R$. 'x' is called pre-image and 'y' is called image of function f. A function can be one to one or many to one but not one to many.
1. Injective / One-to-one function- A function $f: A \to B$ is injective or one-to-one function if for every $b \in B$, there exists at most one $a \in A$ such that $f(s) = t$. This means a function **f** is injective if $a1 \neq a2$ implies $f(a1) \neq f(a2)$
Example
a) $f: N \to N, f(x) = 5x$ is injective.
b) $f: N \to N, f(x) = x2$ is injective.
c) $f: R \to R, f(x) = x2$ is not injective as $(-x)2 = x2$
2. Surjective / onto function - A function $f: A \to B$ is surjective (onto) if the image of f equals its range. Equivalently, for every $b \in B$, there exists some $a \in A$ such that $f(a) = b$. This means that for any y in B, there exists some x in A such that $y = f(x)$

Example

a) $f:N\rightarrow N, f(x)=x+2$ is surjective.

b) $f:R\rightarrow R, f(x)=x2$ is not surjective since we cannot find a real number whose square is negative.

3. Bijective / One-to-one Correspondent - A function $f: A\rightarrow B$ is bijective or one-to-one correspondent if and only if **f** is both injective and surjective.

Problem - Prove that a function $f: R\rightarrow R$ defined by $f(x) = 2x - 3$ is a bijective function.

Explanation − we have to prove this function is both injective and surjective.

If $f(x1) = f(x2)$, then $2x1–3=2x2–3$ and it implies that $x1=x2$. Hence, f is **injective**.

Here, $2x–3=y$

So, $x=(y+5)/3$ which belongs to ' and $f(x) = y$. Hence, f is **surjective**.

Since **f** is both **surjective** and **injective**, we can say **f** is **bijective**.

4. Inverse of a Function - The **inverse** of a one-to-one corresponding function $f: A\rightarrow B$, is the function $g:B\rightarrow A$, holding the following property − $f(x)=y \Leftrightarrow g(y)=x$ The function f is called **invertible**, if its inverse function g exists.

Example

A Function $f:)\rightarrow), f(x)=x+5$, is invertible since it has the inverse function $g:)\rightarrow), g(x)=x-5$.

A Function $f:)\rightarrow), f(x)=x2$ is not invertible since this is not one-to-one as $(-x)2=x2$

5. Composition of Functions - Two functions $f: A\rightarrow B$ and $g: B\rightarrow C$ can be composed to give a composition *gof*. This is a function from A to C defined by $(gof)(x)=g(f(x))$

Example

Let $f(x) = x+2$ and $g(x) = 2x$, find $(fog)(x)$ and $(gof)(x)$

Solution

$(fog)(x)=f(g(x)) = f(2x+1) = 2x+1+2=2x+3$

$(gof)(x)=g(f(x)) = g(x+2) = 2(x+2)+1=2x+5$

Hence, $(fog)(x) \neq (gof)(x)$

6. Recursively Defined Functions – A recursive definition has two parts:

a) Definition of the smallest argument (usually f (0) or f (1)).

b) Definition of f (n), given f (n - 1), f (n - 2) , etc.

f (0) = 1;

f (n) = n . f (n - 1)

Some Facts about Composition

c) If f and g are one-to-one then the function (*gof*) is also one-to-one.

d) If f and g are onto then the function (*gof*) is also onto.

e) Composition always holds associative property but does not hold commutative property.

Pigeonhole Principle

In 1834, German mathematician, Peter Gustav Lejeune Dirichlet, stated a principle which he called the drawer principle. Now, it is known as the pigeonhole principle.

Pigeonhole Principle states that if there are fewer pigeon holes than total number of pigeons and each pigeon is put in a pigeon hole, then there must be at least one pigeon hole with more than one pigeon. If n pigeons are put into m pigeonholes where n > m, there's a hole with more than one pigeon.

Examples

a) Ten men are in a room and they are taking part in handshakes. If each person shakes hands at least once and no man shakes the same man's hand more than once then two men took part in the same number of handshakes.

b) There must be at least two people in a class of 30 whose names start with the same alphabet.

The Inclusion-Exclusion principle

The **Inclusion-exclusion principle** computes the cardinal number of the union of multiple non-disjoint sets. For two sets A and B, the principle states −

$$|A\cup B|=|A|+|B|-|A\cap B|$$

For three sets A, B and C, the principle states −

$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

The generalized formula –

$|\cup ni=1Ai| = \sum 1 \le i < j < k \le n | Ai \cap Aj | + \sum 1 \le i < j < k \le n | Ai \cap Aj \cap Ak | - \cdots + (-1)\, \pi - 1 | A1 \cap \cdots \cap A2 |$

Problem 1

How many integers from 1 to 50 are multiples of 2 or 3 but not both?

Solution

From 1 to 100, there are 50/2=25 numbers which are multiples of 2.

There are 50/3=16 numbers which are multiples of 3.

There are 50/6=8 numbers which are multiples of both 2 and 3.

So, $|A|=25$, $|B|=16$ and $|A \cap B|=8$

$|A \cup B| = |A| + |B| - |A \cap B| = 25+16-8=33$

Problem 2

In a group of 50 students 24 like cold drinks and 36 like hot drinks and each student likes at least one of the two drinks. How many like both coffee and tea?

Solution

Let X is the set of students who like cold drinks and Y is the set of people who like hot drinks.

So, $|X \cup Y|=50$, $|X|=24$, $|Y|=36$

$|X \cap Y| = |X| + |Y| - |X \cup Y| = 24+36-50=60-50=10$

Hence, there are 10 students who like both tea and coffee.

Solved problems on union of sets:

1. Let A = {x: x is a natural number and a factor of 18} and B = {x: x is a natural number and less than 6}. Find A ∪ B.

Solution:

A = {1, 2, 3, 6, 9, and 18} and B = {1, 2, 3, 4, and 5} Therefore, A U B = {1, 2, 3, 4, 5, 6, 9, and 18}

2. Let A = {0, 1, 2, 3, 4, 5}, B = {2, 4, 6, 8} and C = {1, 3, 5, 7} Verify **(A ∪ B) ∪ C = A ∪ (B ∪ C)**

Solution:

(A U B) U C = A U (B U C)

L.H.S. = (A U B) U C

A U B = {0, 1, 2, 3, 4, 5, 6, 8}

(A U B) U C = {0, 1, 2, 3, 4, 5, 6, 7, 8}.....(1)

R.H.S. = A U (B U C)

B U C = {1, 2, 3, 4, 5, 6, 7, 8}

A U (B U C) = {0, 1, 2, 3, 4, 5, 6, 7, 8}.....(2)

Therefore, from (1) and (2), we conclude that;

(A U B) U C = A U (B U C) [verified]

3. Let X = {1, 2, 3, 4}, Y = {2, 3, 5} and Z = {4, 5, 6}.

(i) Verify X ∪ Y = Y ∪ X

(ii) Verify (X ∪ Y) ∪ Z = X ∪ (Y ∪ Z)

Solution:

(i) X U Y = Y U X

L.H.S = X U Y

= {1, 2, 3, 4} U {2, 3, 4}

= {1, 2, 3, 4, 5}

R.H.S. = Y U X

= {2, 3, 5} U {1, 2, 3, 4}

= {2, 3, 5, 1, 4} Therefore, X U Y = Y U X [verified]

(ii) (X ∪ Y) ∪ Z = X ∪ (Y ∪ Z)

L.H.S. = (X U Y) U ZX U Y

= {1, 2, 3, 4} U {2, 3, 5} = {1, 2, 3, 4, 5}

Now (X U Y) U Z= {1, 2, 3, 4, 5, 6} {4, 5, 6}
= {1, 2, 3, 4, 5, and 6}
R.H.S. = X U (Y U Z) Y U Z
= {2, 3, 5} U {4, 5, 6}
= {2, 3, 4, 5, 6}
X U (Y U Z) = {1, 2, 3, 4} U {2, 3, 4, 5, 6}
Therefore, (X U Y) U Z = X U (Y U Z) [verified]

Solved problems on intersection of sets:

1. Let A = {x: x is a natural number and a factor of 18} B = {x: x is a natural number and less than 6} Find A U B and A ∩ B.

Solution:
A = {1, 2, 3, 6, 9, 18}
B = {1, 2, 3, 4, 5}
Therefore, A ∩ B = {1, 2, 3}

2. If P = {multiples of 3 between 1 and 20} and Q = {even natural numbers upto 15}. Find the intersection of the two given set P and set Q.

Solution:
P = {multiples of 3 between 1 and 20} So, P = {3, 6, 9, 12, 15, 18}
Q = {even natural numbers upto 15} So, Q = {2, 4, 6, 8, 10, 12, 14}
Therefore, intersection of P and Q is the largest set containing only those elements which are common to both the given sets P and Q
Hence, P ∩ Q = {6, 12}.

3. Let A = {0, 1, 2, 3, 4, 5}, B = {2, 4, 6, 8} and C = {1, 3, 5, 7} Verify (A ∩ B) ∩ C = A ∩ (B ∩ C)

Solution:
(A ∩ B) ∩ C = A ∩ (B ∩ C)
L.H.S. = (A ∩ B) ∩ CA ∩ B
= {2, 4}(A ∩ B) ∩ C = {Ø} ..... (1)
'.H.S. = A ∩ (B ∩ C)B ∩ C = {Ø}
A ∩ {B ∩ C} = {Ø}..... (2)
Therefore, from (1) and (2), we conclude that;(A ∩ B) ∩ C = A ∩ (B ∩ C) [verified]

4. Given three sets P, Q and R such that:
P = {x: x is a natural number between 10 and 16},
Q = {y: y is an even number between 8 and 20} and
R = {7, 9, 11, 14, 18, 20}

a. Find the difference of two sets P and Q
b. Find Q - R
c. Find R - P
d. Find (Q – P)

Solution:
According to the given statements:
P = {11, 12, 13, 14, 15}
Q = {10, 12, 14, 16, 18}
R = {7, 9, 11, 14, 18, 20}
(i) P – Q = {those elements of set P which are not in set Q}
= {11, 13, and 15}
(ii) Q – R = {those elements of set Q not belonging to set R}
= {10, 12, and 16}
(iii) R – P = {those elements of set R which are not in set P}
= {7, 9, 18, and 20}
(iv) Q – P = {those elements of set Q not belonging to set P}

= {10, 16, and 18}

Problems on Operation on Sets

1. If A = {1, 3, 5}, B = {3, 5, 6} and C = {1, 3, 7}

(i) Verify that A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)

(ii) Verify A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)

Solution:

(i) A U (B ∩ C) = (A U B) ∩ (A U C)

L.H.S. = A U (B ∩ C)

B ∩ C = {3}

A U (B ∩ C) = {1, 3, 5} U {3} = {1, 3, 5} .....(1)

R.H.S. = (A U B) ∩ (A U C)

A U B = {1, 3, 5, 6}

A U C = {1, 3, 5, 7}

(A U B) ∩ (A U C) = {1, 3, 5, 6} ∩ {1, 3, 5, 7} = {1, 3, 5} ..... (2)

From (1) and (2), we conclude that;

A U (B ∩ C) = A U B ∩ (A U C)[*verified*]

(ii) A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)

L.H.S. = A ∩ (B U C)

B U C = {1, 3, 5, 6, 7}

A ∩ (B U C) = {1, 3, 5} ∩ {1, 3, 5, 6, 7} = {1, 3, 5} ..... (1)

'.H.S. = (A ∩ B) U (A ∩ C)

A ∩ B = {3, 5}

A ∩ C = {1, 3}

(A ∩ B) U (A ∩ C) = {3, 5} U {1, 3} = {1, 3, 5}.....(2)

From (1) and (2), we conclude that;

A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)[*verified*]

2. Let A = {a, b, d, e}, B = {b, c, e, f} and C = {d, e, f, g}

(i) Verify A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)

(ii) Verify A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)

Solution:

(i) A ∩ (B U C) = (A ∩ B) U (A ∩ C)

L.H.S. = A ∩ (B U C)

B U C = {b, c, d, e, f, g}

A ∩ (B U C) = {b, d, e}..... (1)

R.H.S. = (A ∩ B) U (A ∩ C)

A ∩ B = {b, e}

A ∩ C = {d, e}

(A ∩ B) U (A ∩ C) = {b, d, e}..... (2)

From (1) and (2), we conclude that;

A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C) [*verified*]

(ii) A U (B ∩ C) = (A U B) ∩ (A U C)

L.H.S. = A U (B ∩ C)

B ∩ C = {e, f}

A U (B ∩ C) = {a, b, d, e, f} .....(1)

R.H.S. = (A U B) ∩ (A U C)

AUB = {a, b, c, d, e, f}

AUC = {a, b, d, e, f, g}

(A U B) ∩ (A U C) = {a, b, d, e, f}..... (2)

From (1) and (2), we conclude that;

A U (B ∩ C) = A U B ∩ (A U C) [*verified*]

Numerical on Sets:-

1. Let A and B be two finite sets such that n (A) = 20, n (B) = 28 and n (A ∪ B) = 36, find n (A ∩ B).

Solution:

Using the formula n (A ∪ B) = n (A) + n (B) - n (A ∩ B).

Then n (A ∩ B) = n (A) + n (B) - n (A ∪ B)

= 20 + 28 - 36

= 48 - 36

= 12

2. If n (A - B) = 18, n (A ∪ B) = 70 and n (A ∩ B) = 25, then find n (B).

Solution:

Using the formula n(A∪B) = n(A - B) + n(A ∩ B) + n(B - A)

70 = 18 + 25 + n(B - A)

70 = 43 + n(B - A)

n(B - A) = 70 - 43

n(B - A) = 27

Now n(B) = n(A ∩ B) + n(B - A)

= 25 + 27

= 52

3. In a group of 60 people, 27 like cold drinks and 42 like hot drinks and each person likes at least one of the two drinks. How many like both coffee and tea?

Solution:

Let A = Set of people who like cold drinks.

B = Set of people who like hot drinks.

Given

(A ∪ B) = 60 n (A) = 27 n (B) = 42 then;

n (A ∩ B) = n(A) + n(B) - n(A ∪ B)

= 27 + 42 - 60

= 69 - 60 = 9

Therefore, 9 people like both tea and coffee.

4. There are 35 students in art class and 57 students in dance class. Find the number of students who are either in art class or in dance class.

(i) When two classes meet at different hours and 12 students are enrolled in both activities.

(ii) When two classes meet at the same hour.

Solution:

n(A) = 35, n(B) = 57, n(A ∩ B) = 12

(Let A be the set of students in art class.

B be the set of students in dance class.)

(i) When 2 classes meet at different hours n(A ∪ B) = n(A) + n(B) - n(A ∩ B)

= 35 + 57 - 12

= 92 - 12

= 80

(ii) When two classes meet at the same hour, A∩B = Ø n (A ∪ B) = n(A) + n(B) - n(A ∩ B)

= n(A) + n(B)

= 35 + 57

= 92

5. In a group of 100 persons, 72 people can speak English and 43 can speak French. How many can speak English only? How many can speak French only and how many can speak both English and French?
Solution:

Let A be the set of people who speak English.
B is the set of people who speak French.
A - B is the set of people who speak English and not French.
B - A be the set of people who speak French and not English.
A ∩ B be the set of people who speak both French and English.
Given,
n (A) = 72 n (B) = 43 n (A ∪ B) = 100
Now, n (A ∩ B) = n (A) + n (B) - n (A ∪ B)
= 72 + 43 - 100
= 115 - 100
= 15
Therefore, Number of persons who speak both French and English = 15
n(A) = n(A - B) + n(A ∩ B)
⇒ n(A - B) = n(A) - n(A ∩ B)
= 72 - 15
= 57
n(B - A) = n(B) - n(A ∩ B)
= 43 - 15
= 28
Therefore, Number of people speaking English only = 57
Number of people speaking French only = 28

6. In a competition, a school awarded medals in different categories. 36 medals in dance, 12 medals in dramatics and 18 medals in music If these medals went to a total of 45 persons and only 4 persons got medals in all the three categories, how many received medals in exactly two of these categories?
Solution:

Let A = set of persons who got medals in dance.
B = set of persons who got medals in dramatics.
C = set of persons who got medals in music.
Given,
n(A) = 36 n(B) = 12 n(C) = 18
n(A ∪ B ∪ C) = 45 n(A ∩ B ∩ C) = 4
We know that number of elements belonging to exactly two of the three sets A, B, C
= n(A ∩ B) + n(B ∩ C) + n(A ∩ C) - 3n(A ∩ B ∩ C)
= n(A ∩ B) + n(B ∩ C) + n(A ∩ C) - 3 × 4.....(i)
n(A ∪ B ∪ C) = n(A) + n(B) + n(C) - n(A ∩ B) - n(B ∩ C) - n(A ∩ C) + n(A ∩ B ∩ C)
Therefore, n(A ∩ B) + n(B ∩ C) + n(A ∩ C) = n(A) + n(B) + n(C) + n(A ∩ B ∩ C) - n(A ∪ B ∪ C)
From (i) required number
= n(A) + n(B) + n(C) + n(A ∩ B ∩ C) - n(A ∪ B ∪ C) - 12
= 36 + 12 + 18 + 4 - 45 - 12
= 70 - 67
= 3

7. Each student in a class of 40 plays at least one indoor game, chess, carom and scrabble. 18 play chess, 20 play scrabble and 27 play carom. 7 play chess and scrabble, 12 play scrabble and carom and 4 play chess, carom and scrabble. Find the number of students who play (i) chess and carom. (ii) Chess, carom but not scrabble.

Solution:
Let A be the set of students who play chess
B be the set of students who play scrabble
C be the set of students who play carom
Therefore, We are given n(A U B U C) = 40,
n(A) = 18, n(B) = 20 n(C) = 27,
n(A ∩ B) = 7, n(C ∩ B) = 12 n(A ∩ B ∩ C) = 4
We have
n (A U B U C) = n(A) + n(B) + n(C) - n(A ∩ B) - n(B ∩ C) - n(C ∩ A) + n(A ∩ B ∩ C)
Therefore, 40 = 18 + 20 + 27 - 7 - 12 - n(C ∩ A) + 4
40 = 69 – 19 - n(C ∩ A)
40 = 50 - n(C ∩ A) n(C ∩ A) = 50 - 40
n (C ∩ A) = 10
Therefore, Number of students who play chess and carom are 10
Also, number of students who play chess, carom and not scrabble
= n(C ∩ A) - n(A ∩ B ∩ C)
= 10 – 4
= 6

Mathematical Induction

Mathematical induction is a technique for proving results or establishing statements for natural numbers. This part illustrates the method through a variety of examples.

Definition - Mathematical Induction is a mathematical technique which is used to prove a statement, a formula or a theorem is true for every natural number.

The technique involves two steps to prove a statement, as stated below −

Step 1(Base step) − It proves that a statement is true for the initial value.

Step 2(Inductive step) − It proves that if the statement is true for the n^{th} iteration (or number *n*), then it is also true for $(n+1)^{th}$ iteration (or number *n+1*).

How to Do It

Step 1 − Consider an initial value for which the statement is true. It is to be shown that the statement is true for n = initial value.

Step 2 − Assume the statement is true for any value of *n = k*. Then prove the statement is true for *n = k+1*. We actually break *n = k+1* into two parts, one part is *n = k* (which is already proved) and try to prove the other part.

1. $3n-1$ is a multiple of 2 for n = 1, 2, 3.....

Solution

Step 1 − for $n=1$, $3*1-1=3-1=2$ which is a multiple of 2

Step 2 − Let us assume $3n-1$ is true for $n=k$, Hence, $3k-1$ is true (It is an assumption)

We have to prove that $3k+1-1$ is also a multiple of 2

$3k+1-1=3\times3k-1=(2\times3k)+(3k-1)$

The first part $(2\times3k)$ is certain to be a multiple of 2 and the second part $(3k-1)$ is also true as our previous assumption.

Hence, $3k+1-1$ is a multiple of 2.

So, it is proved that $3n-1$ is a multiple of 2.

2. $1+3+5+.+(2n-1)=n^2$ for $n=1, 2, 3…$

Solution

Step 1 − for $n=1$, $1=1^2$, Hence, step 1 is satisfied.

Step 2 − Let us assume the statement is true for $n=k$

Hence, $1+3+5+\cdots+(2k-1)=k^2$ is true (It is an assumption)

We have to prove that $1+3+5+.+(2(k+1)-1)=(k+1)^2$ also hold

$1+3+5+\cdots+(2(k+1)-1)$
$=1+3+5+\cdots+(2k+2-1)$
$=1+3+5+\cdots+(2k+1)$
$=1+3+5+\cdots+(2k-1)+(2k+1)$
$=k^2+(2k+1)$
$=(k+1)^2$
So, $1+3+5+\cdots+(2(k+1)-1)=(k+1)^2$ hold which satisfies the step 2.
Hence, $1+3+5+\cdots+(2n-1)=n2$ is proved.

3. **Prove that $(ab)^n = a^n b^n$ is true for every natural number *n***
Solution
Step 1 − For $n=1$, $(ab)^1=a^1b^1=ab$, Hence, step 1 is satisfied.
Step 2 − Let us assume the statement is true for $n=k$, Hence, $(ab)^k=a^kb^k$ is true (It is an assumption).
We have to prove that $(ab)^{k+1}=a^{k+1}b^{k+1}$ also hold
Given, $(ab)^k = a^kb^k$ Or, $(ab)^k(ab)=(a^kb^k)(ab)$
[Multiplying both side by 'ab']
Or, $(ab)^{k+1}= (a\ a^k)\ (b\ b^k)$
Or, $(ab)^{k+1}= (a^{k+1}b^{k+1})$
Hence, step 2 is proved.
So, $(ab)^n = a^nb^n$ is true for every natural number n.

Strong Induction
Strong Induction is another form of mathematical induction. Through this induction technique, we can prove that a propositional function, $P(n)$ is true for all positive integers, n , using the following steps −
Step 1(Base step) − It proves that the initial proposition $P(1)$ true.
Step 2(Inductive step) − It proves that the conditional statement $[P(1) \wedge P(2) \wedge P(3) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for positive integers *k.*

Proof by Contradiction. To prove a statement P is true, we begin by assuming P false and show that this leads to a contradiction; something that always false.
Many of the statements we prove have the form $P \Rightarrow Q$ which, when negated, has the form $P \Rightarrow \sim Q$.

Proposition $P \Rightarrow Q$

Proof: Assume, for the sake of contradiction P is true but Q is false. · · · Since we have a contradiction, it must be that Q is true

Proof: √ 2 is irrational

Proof: Suppose √ 2 is rational. Then integers a and b exist so that √ 2 = a/b. Without loss of generality we can assume that a and b have no factors in common (i.e., the fraction is in simplest form). Multiplying both sides by b and squaring, we have
$2b^2 = a^2$ so we see that a^2 is even. This means that a is even (how would you prove this?) so a = 2m for some m ∈). Then
$2b^2 = a^2 = (2m)^2 = 4m^2$ which, after dividing by 2, gives $b^2 = (2m)^2$ so b^2 is even. This means b = 2n for some n ∈).

Subject Notes

UNIT-2

Group Theory is a branch of mathematics and abstract algebra that defines an algebraic structure named as **Group**. Generally, a group comprises of a set of elements and an operation over any two elements on that set to form a third element also in that set. In 1854, Arthur Cayley, the British Mathematician, gave the modern definition of group for the first time −

"**A set of symbols all of them different, and such that the product of any two of them (no matter in what order), or the product of any one of them into itself, belongs to the set, is said to be a group. These symbols are not in general convertible [commutative], but are associative."**

Any set of elements in a mathematical system may be defined with a set of operators and a number of postulates.

A **binary operator** defined on a set of elements is a rule that assigns to each pair of elements a unique element from that set. For example, given the set A={1,2,3,4,5}

, we can say ⊗ is a binary operator for the operation $c=a⊗b$, if it specifies a rule for finding c for the pair of (a,b), such that a,b,$c∈A$.

The **postulates** of a mathematical system form the basic assumptions from which rules can be deduced. The postulates are −

1. **Closure-** A set is closed with respect to a binary operator if for every pair of elements in the set; the operator finds a unique element from that set.

Example

Let A={0,1,2,3,4,5,…}

This set is closed under binary operator into (∗), because for the operation $c=a∗b$, for any a,$b∈A$, the product $c∈A$.

The set is not closed under binary operator divide (÷), because, for the operation $c=a÷b$, for any a,$b∈A$, the product c may not be in the set A. If a=7,b=2, then c=3.5. Here a,$b∈A$ but $c∉A$

2. **Associative Laws -** A binary operator ⊗ on a set A is associative when it holds the following property − $(x⊗y)⊗z=x⊗(y⊗z)$, where x,y,$z∈A$

Example

Let A={1,2,3,4} The operator plus (+) is associative because for any three elements, x,y,$z∈A$, the property $(x+y)+z=x+(y+z)$ holds.

The operator minus (−) is not associative since $(x−y)−z≠x−(y−z)$

3. **Commutative Laws -** A binary operator ⊗ on a set A is commutative when it holds the following property −

$x⊗y=y⊗x$, where x,$y∈A$

Example

Let A={1,2,3,4} The operator plus (+) is commutative because for any two elements, x,$y∈A$, the property $x+y=y+x$ holds.

The operator minus (−) is not associative since $x−y≠y−x$

4. **Distributive Laws -** Two binary operators ⊗ and ⊛ on a set A, are distributive over operator ⊛ when the following property holds −

$x⊗(y⊛z)=(x⊗y)⊛(x⊗z)$, where x,y,$z∈A$

Example

Let A={1,2,3,4} The operators into (∗) and plus (+) are distributive over operator + because for any three elements, x,y,$z∈A$, the property $x∗(y+z)=(x∗y)+(x∗z)$ holds.

However, these operators are not distributive over ∗ since $x+(y∗z)≠(x+y)∗(x+z)$

5. **Identity Element -** A set A has an identity element with respect to a binary operation ⊗ on A, if there exists an element $e∈A$, such that the following property holds − $e⊗x=x⊗e$, where $x∈A$

Example

Let Z={0,1,2,3,4,5,…} The element 1 is an identity element with respect to operation $*$
since for any element $x \in Z$,
$1*x=x*1$
On the other hand, there is no identity element for the operation minus (−)

6. **Inverse -** If a set A has an identity element e with respect to a binary operator $\otimes$, it is said to have an inverse whenever for every element $x \in A$, there exists another element $y \in A$, such that the following property holds $-\boldsymbol{x \otimes y=e}$

Example

Let A={⋯−4,−3,−2,−1,0,1,2,3,4,5,…} Given the operation plus (+) and e=0, the inverse of any element x is $(-x)$ since $x+(x)=0$

Semigroup

A finite or infinite set 'S' with a binary operation 'o' (Composition) is called semigroup if it holds following two conditions simultaneously −

i. **Closure** − For every pair $(a,b) \in S,(aob)$ has to be present in the set S
ii. **Associative** − For every element $a,b,c \in S,(aob)oc=ao(boc)$ must hold.

Example

The set of positive integers (excluding zero) with addition operation is a semigroup. For example, S={1,2,3,…}

Here closure property holds as for every pair $(a,b) \in S,(a+b)$ is present in the set S. For example, $1+2=3 \in S$]
Associative property also holds for every element $a,b,c \in S,(a+b)+c=a+(b+c)$. For example, (1+2)+3=1+(2+3)=5

Monoid

A monoid is a semigroup with an identity element. The identity element (denoted by e or E) of a set S is an element such that $(aoe)=a$, for every element $a \in S$. An identity element is also called a **unit element**. So, a monoid holds three properties simultaneously − **Closure, Associative, Identity element**.

Example

The set of positive integers (excluding zero) with multiplication operation is a monoid. S={1,2,3,…}

1. Here closure property holds as for every pair $(a,b) \in S,(a \times b)$ is present in the set S. [For example, $1 \times 2=2 \in S$ and so on]
2. Associative property also holds for every element $a,b,c \in S,(a \times b) \times c=a \times (b \times c)$ [For example, $(1 \times 2) \times 3=1 \times (2 \times 3)=6$ and so on]
3. Identity property also holds for every element $a \in S,(a \times e)=a$ [For example, $(2 \times 1)=2,(3 \times 1)=3$

and so on]. Here identity element is 1.

Group

A group is a monoid with an inverse element. The inverse element (denoted by I) of a set S is an element such that $(aoI)=(Ioa)=a$, for each element $a \in S$. So, a group holds four properties simultaneously –

i) Closure, ii) Associative, iii) Identity element, iv) Inverse element.

The order of a group G is the number of elements in G and the order of an element in a group is the least positive integer n such that an is the identity element of that group G.

Examples

The set of $N \times N$ nonsingular matrices form a group under matrix multiplication operation.

1. The product of two $N \times N$ nonsingular matrices is also an $N \times N$
2. non-singular matrix which holds closure property.
3. Matrix multiplication itself is associative. Hence, associative property holds.
4. The set of $N \times N$ nonsingular matrices contains the identity matrix holding the identity element property.

As all the matrices are nonsingular they all have inverse elements which are also nonsingular matrices. Hence,

inverse property also holds.

AbelianGroup

An abelian group G is a group for which the element pair $(a,b) \in G$ always holds commutative law. So, a group holds five properties simultaneously –

i) Closure, ii) Associative, iii) Identity element, iv) Inverse element, v) Commutative.

Example

The set of positive integers (including zero) with addition operation is an abelian group. $G=\{0,1,2,3,\ldots\}$

1. Here closure property holds as for every pair $(a,b) \in S,(a+b)$ is present in the set S. [For example, $1+2=2 \in S$ and so on]
2. Associative property also holds for every element $a,b,c \in S,(a+b)+c=a+(b+c)$ [For example, (1+2)+3=1+(2+3)=6 and so on]
3. Identity property also holds for every element $a \in S,(a \times e)=a$ [For example, (2×1)=2,(3×1)=3 and so on]. Here, identity element is 1.
4. Commutative property also holds for every element $a \in S,(a \times b)=(b \times a)$ [For example, (2×3)=(3×2)=3and so on]

Permutation Group : Let S be a set. A permutation of S is simply a bijectionf : S −→ S.

Lemma 2.1

Let S be a set.

(1) Let f and g be two permutations of S. Then the composition of f and g is a permutation of S.

(2) Let f be a permutation of S. Then the inverse off is a permutation of S.

Lemma 2.2

Let S be a set. The set of all permutations, under the operation of composition of permutations, forms a group A(S).

Proof : We know that

- the set of permutations is closed under composition of functions.
- composition of functions is associative

We check the three axioms for a group.

Let i: S −→ S be the identity function from S to S.

Let f be a permutation of S.

Clearly f ◦ i = i ◦ f = f.

Thus i acts as an identity. Let f be a permutation of S.

Then the inverse g of f is a permutation of S and f ◦ g = g ◦ f = i, by definition. Thus inverses exist and G is a group

Products of permutations: In order to form the product of permutations σwe mean apply σ first and then apply . In terms of the cycle representation this means that products are carried out reading from left to right. This assumes that you are now writing functions on the right! To carry out a product we construct cycles as described above but with one difference. Select an element i ∈ A. The next element in the cycle is iσ followed by $(\sigma)^2$ etc. Repeat this procedure until every element of *A* appears in some cycle, again single cycles are suppressed.

To illustrate σ=(1 4 5) (2 3) this, if and then= (2 4) (5 1)

σ = (1 4 5) (2 3) (2 4) (5 1) =(1 2 3 4)

σ^2 = (1 4 5) (2 3) (1 4 5) (2 3) = (1 5 4)

Another way to carry out this multiplication process (although no different from what has already been said) is as follows. Write down your starting element. Read the cycles from left to right. If the starting element is in a cycle read its successor (you may have to wrap around to the beginning of the cycle). This is you new number. Now go to the next cycle in which this new number appears. Read its successor and this becomes the new number. Repeat until you reach the end of the cycles (you have reached the rightmost end). This is now written as the successor of the starting element. Now this number becomes the new starting element. Repeat the above to find its successor. When you obtain a successor which is the original starting element, close off your cycle. Open up a new cycle and start all over again until all numbers have appeared in exactly one cycle. Now erase cycles of length one.

Cyclic Group and Subgroup

A **cyclic group** is a group that can be generated by a single element. Every element of a cyclic group is a power of some specific element which is called a generator. A cyclic group can be generated by a generator 'g', such that every other element of the group can be written as a power of the generator 'g'.

Example

The set of complex numbers {1,−1,*i*,−*i*} under multiplication operation is a cyclic group.

Solution- There are two generators – *I* and –*i* as *i*1=*i*,*i*2=−1,*i*3=−*i*,*i*4=1 and also (–*i*)1=−*i*,(–*i*)2=−1,(–*i*)3=*i*,(–*i*)4=1 which covers all the elements of the group. Hence, it is a cyclic group.

Note − A **cyclic group** is always an abelian group but not every abelian group is a cyclic group. The rational numbers under addition is not cyclic but is abelian.

A **subgroup** H is a subset of a group G (denoted by *H*≤*G*) if it satisfies the four properties simultaneously − **Closure, Associativity, Identity element**, and **Inverse**.

A subgroup H of a group G that does not include the whole group G is called a proper subgroup (Denoted by *H*<*G*). A subgroup of a cyclic group is cyclic and aabelian subgroup is also abelian.

Example

Let a group *G*={1,*i*,−1,−*i*}

Then some subgroups are *H*1={1},*H*2={1,−1}, This is not a subgroup − *H*3={1,*i*} because that (*i*)−1=−*i* is not in *H*3

Normal Subgroup: Let H be a subgroup of a group G. The similarity transformation of H by a fixed element x in G not in H always gives a subgroup. If xHx^(-1)=H for every element x in G, then H is said to be a normal subgroup of G, written H<|G.

Normal subgroups are also known as invariant subgroups or self-conjugate subgroup.All subgroups of Abelian groups are normal.

Coset: Given H ≤ G, a left coset of H in G is a subset of G of the form gH = {gh|h∈ H} for some g ∈ G. Similarly a right coset of H in G is a subset of G of the form Hg = {hg|h∈ H} for some g ∈ G. Notice since g = eg = ge that g ∈ Hg and g ∈gH.

Example Suppose G = Σ_3, H =< (1, 2) >= {e,(1, 2)} and g = (1, 3). Then a simple computation shows that gH = {(1, 3),(1, 2, 3)} while Hg = {(1, 3),(1, 3, 2)} and so gH 6= Hg. Thus we see that for a fixed element g, the left cosetgH may be different from the right coset Hg in general.

Multiplying elements and sets Of course, the expression gH does not make immediate sense from the group axioms. What it means, by definition, is gH = {gh | h ∈ H} .

To put this another way, the golden rule is this: if you know that f ∈gH, then you can conclude that there is some h ∈ H so that f = gh.

Applying the golden rule Consider G = S4 and H = {id,(1, 2)}. If g = (2, 3, 4), then gH = {(2, 3, 4),(2, 3, 4)(1, 2)} = {(2, 3, 4),(1, 3, 4, 2)}. Now let f = (3, 4, 2, 1)—this is an element of gH. Which h ∈ H satisfies f = gh?

Or

if g = (1, 3)(2, 4), then gH = {(1, 3)(2, 4),(1, 4, 2, 3)}. If you let f = (1, 4, 2, 3), which h ∈ H satisfies f = gh this time?

compute the two cosets g1H ⊂ S4 and g2H ⊂ S4 for H = {id,(1, 2, 3, 4),(1, 3)(2, 4),(1, 4, 3, 2)} and g1 = (1, 3, 2), g2 = (1, 2, 3, 4).

Homomorphism

Definition: A group homomorphism ϕ : G → G0 is an isomorphism if ϕ is a bijection. If there is an isomorphism between G and G0 we say G and G0 are isomorphic. This is denoted by G ∼= G0 .

Given a homomorphism ϕ : G → G0 there are subgroups of each that can indicate to us whether ϕ is injective or surjective. Definition. Let ϕ : G → G0 be a homomorphism. Define ker(ϕ) = {g ∈ G : ϕ(g) = eG0}. This is called the kernel of ϕ. Define im(ϕ) = {ϕ(g) : g ∈ G}. This is called the image of ϕ. We usually use the notation ϕ(G) for im(G).

Example. If ϕ : GL2(ʻ) → ʻ\{0} is given by ϕ(A) = det(A) then ker(ϕ(A)) = SL2(ʻ) and im(ϕ) = ʻ\{0}.

Theorem Letϕ : G → G0 be a homomorphism. Then 1.ker(ϕ) is a subgroup of G, and ϕ is injective if and only if ker(ϕ) = eG. 2. im(ϕ) is a subgroup of G0 , and ϕ is surjective if and only if im(ϕ) = G0 (or equivalently, ϕ(G) = G0).

Proof. If a, b ∈ker(ϕ), then ϕ(ab−1) = ϕ(a)ϕ(b) −1 = eG0(eG0) −1 = eG0 so by the Subgroup Test, ker(ϕ) is a subgroup.

Now if ker(ϕ) = {e} then ϕ(a) = ϕ(b) ⟹ϕ(ab−1) = e ⟹ ab−1 = e ⟹ a = b.

Moreover, if ϕ is injective, then

ϕ(a) = e ⟹ϕ(a) = ϕ(e) ⟹ a = e, so ker(ϕ) = {e}.

Isomorphism: The homomorphism ϕ : G → G0 is an isomorphism if and only if there exists a homomorphism ψ : G0 → G such that ϕ ◦ ψ = ψ ◦ ϕ are identity maps on their respective groups.

Proof : Define ψ(a) to be the unique pre-image of a under ϕ. Since ϕ is a bijection, this is well defined and ϕ ◦ ψ = ψ ◦ ϕ are identity maps between their respective groups. One needs to check ψ is indeed a homomorphism, but this effectively comes for free since ϕ is one.

Example and standard result on Group

Integers Z with addition

(G1) a, b ∈ Z ⟹ a + b ∈ Z

(G2) (a + b) + c = a + (b + c)

(G3) the identity element is 0 as a + 0 = 0 + a = a and 0 ∈ Z

(G4) the inverse of a ∈ Z is −a as a + (−a) = (−a) + a = 0 and −a ∈ Z (G5) a + b = b + a

The set Zn of congruence classes modulo n with addition

(G1) [a], [b] ∈ Zn ⟹ [a] + [b] = [a + b] ∈ Zn

(G2) ([a] + [b]) + [c] = [a + b + c] = [a] + ([b] + [c])

(G3) the identity element is [0] as [a] + [0] = [0] + [a] = [a]

(G4) the inverse of [a] is [−a] as [a] + [−a] = [−a] + [a] = [0]

(G5) [a] + [b] = [a + b] = [b] + [a]

The set Gn of invertible congruence classes modulo n with multiplication

A congruence class [a]n ∈ Zn belongs to Gn if gcd(a, n) = 1.

(G1) [a]n, [b]n ∈Gn =⇒gcd(a, n) = gcd(b, n) = 1 =⇒gcd(ab, n) = 1 =⇒ [a]n[b]n = [ab]n ∈Gn
(G2) ([a][b])[c] = [abc] = [a]([b][c])
(G3) the identity element is [1] as [a][1] = [1][a] = [a]
(G4) the inverse of [a] is [a] −1 by definition of [a] −1
(G5) [a][b] = [ab] = [b][a]

Permutations S(n) with composition (= multiplication)
(G1) π and σ are bijective functions from the set {1, 2, . . . , n} to itself =⇒ so is πσ
(G2) (π σ)τ and π(σ τ) applied to k, 1 ≤ k ≤ n, both yield π(σ(τ (k))).
(G3) the identity element is id as π id = id π = π
(G4) the inverse of π is π −1 by definition of the inverse function (G5) fails for n ≥ 3 as (as (1 2)(2 3) = (1 2 3) while (2 3)(1 2) = (1 3 2).

The definition of a ring: A structure (R, +, ·) is a ring if R is a non-empty set and + and · are binary operations:
+: ‘ × ‘ → ‘, (a, b) 7→ a + b · : ‘ × ‘ → ‘, (a, b) 7→ a · b
such that
Addition: (R, +) is an abelian group, that is,
(A1) associativity: for all a, b, c ∈ R we have a + (b + c) = (a + b) + c
(A2) zero element: there exists 0 ∈ R such that for all a ∈ R we have a + 0 = 0 + a = a
(A3) inverses: for any a ∈ R there exists −a ∈ R such that a + (−a) = (−a) + a = 0
(A4) commutativity: for all a, b ∈ R we have a + b = b + a

Multiplication:
(M1) associativity: for all a, b, c ∈ R we have a · (b · c) = (a · b) · c

Addition and multiplication together (D) for all a, b, c ∈ R,

a · (b + c) = a · b + a · c and (a + b) · c = a · b + b · c.
We sometimes say ‘‘ is a ring’, taken it as given that the ring operations are denoted + and ·. As in ordinary arithmetic we shall frequently suppress · and write ab instead of a · b

Special types of rings: definitions. Assume (R; +, ·) is a ring. We say R is a commutative ring if its multiplication · is commutative, that is,
(M4) Commutativity: a · b = b · a for all a, b ∈ R. We say R is a ring with 1 (or ring with identity) if there exists an identity for multiplication, that is,
(M2) identity element: there exists 1 ∈ R such that for all a ∈ R we have a · 1 = 1 · a = a.

Examples of rings
Number systems
(1) All of Z, Q, R and C are commutative rings with identity (with the number 1 as the identity).
(2) N is NOT a ring for the usual addition and multiplication. These are binary operations and we do have a zero element, namely 0, so axiom (A2) holds. However (A3) (existence of additive inverses) fails: there is no n ∈ N for which 1 + n = 0, for example.
(3) Consider the set of even integers, denoted 2Z, with the usual addition and multiplication. This is a commutative ring without an identity. To verify that (M2) fails it is not sufficient just to say that the integer 1 does not belong to 2Z. Instead we argue as follows. Suppose for contradiction that there were an element e ∈ 2Z such that n·e = n for all n ∈ 2Z. In particular 2e = 2, from which we deduce that e would have to be 1. Since 1 ∈/ 2Z we have a contradiction.

Matrix rings Under the usual matrix addition and multiplication Mn(R) and Mn(C), are rings with 1, but are not commutative (unless n = 1). If we restrict to invertible matrices we no longer have a ring, because there is then no zero for addition.

PolynomialsPolynomials, with real coefficients, form a commutative ring with identity under the usual addition and multiplication; we denote this by R[x].

Modular arithmetic Binary arithmetic on {0, 1} (see 1.2(4)) gives us a 2-element commutative ring with identity. More generally we get a commutative ring with identity if we consider addition and multiplication mod n on {0, 1, . . . , n − 1}.

Calculational rules for rings.

Assume that (R; +, ·) is a commutative ring.

Let a, b, c ∈ R.

(i) If a + b = a + c then b = c.

(ii) If a + a = a then a = 0.

(iii) −(−a) = a.

(iv) 0a = 0.

(v) −(ab) = (−a)b = a(−b). Assume in addition that ' has an identity 1 Then

(vi) (−1)a = −a.

(vii) If a ∈ R has a multiplicative identity a −1 then ab = 0 implies b = 0.

Field: a field is a ring in which the elements, other than the identity element for addition, and the multiplication operator, also form a group.

- There are only two kinds of finite fields. One kind is the field formed by addition and multiplication modulo a prime number.
- The other kind of finite field has a number of elements that is a power of a prime number.
- The addition operator consists of multiple independent additions modulo that prime. The elements of the field can be thought of as polynomials whose coefficients are numbers modulo that prime. In that case, multiplication is polynomial multiplication, where not only the coefficients modulo that prime, but the polynomials are modulo a special kind of polynomial, known as a primitive polynomial. All finite fields, but particularly those of this second kind, are known as Galois fields.
- A commutative ring which has more than one element such that every non-zero element of S has a multiplicative inverse in S is called a field.

The ring of even integers is a subring of the ring of integers. Let •> and Å , ¤ > be rings. A mapping of g : '® S is called a ring homomorphism from and Å ,¤ > if for any a, b, Î R g(a + b) = g(a) Å g(b) and g(a • b) = g(a) ¤ g(b).

Standard results

If R is a ring and a, b, c, d ∈ R, evaluate (a + b)(c + d).

Solution: (a + b)(c + d) = a(c + d) + b(c + d)

by distributive law

= (ac + ad) + (bc + bd)

= ac + ad + bc + bd

Prove that if a, b ∈ R, then $(a + b)^2 = a^2 + ab + ba + b^2$ where by x^2 we mean xx.

Solution: $(a + b)^2 = (a + b)(a + b) = a(a + b) + b(a + b) = a^2 + ab + ba + b^2$ Note that if R is not a commutative ring ab 6= ba.

If in a ring R every x ∈ R satisfies $x^2 = x$,prove that R must be commutative (A ring in which $x^2 = x$ for all elements is called a Boolean ring).

Solution: Let x, y ∈ R. Then $(x + y)^2 = (x + y)(x + y) = x^2 + xy + yx + y^2$ Since $x^2 = x$ and $y^2 = y$ we have x + y = x + xy + yx + y. Hence xy = −yx. But for every x ∈ R $(−x) = (−x)^2 = (−x)(−x) = x^2 = x$. Hence −yx = yx i.e. we obtain xy = yx.

Prove that any field is an integral domain.

Solution: Let a 6= 0 and b be two elements in the field F and ab = 0. Since F is a field and a 6= 0. we have a −1 ∈ F. Hence a −1ab = a −10 = 0. So we obtain b = 0. Hence there exists no zero divisor in F.

If U is an ideal of R and 1 ∈ U, prove that U = R.

Solution: Since for any r ∈ R and u ∈ U, ru∈ U we have for any r ∈ R, r1 = r ∈ U. Hence R = U.

UNIT-3

Propositional Logic

The rules of mathematical logic specify methods of reasoning mathematical statements. Greek philosopher, Aristotle, was the pioneer of logical reasoning. Logical reasoning provides the theoretical base for many areas of mathematics and consequently computer science. It has many practical applications in computer science like design of computing machines, artificial intelligence, definition of data structures for programming languages etc.

Propositional Logic is concerned with statements to which the truth values, “true” and “false”, can be assigned. The purpose is to analyze these statements either individually or in a composite manner.

Prepositional Logic – Definition

A proposition is a collection of declarative statements that has either a truth value "true” or a truth value "false". A propositional consists of propositional variables and connectives. We denote the propositional variables by capital letters (A, B, etc). The connectives connect the propositional variables.

Some examples of Propositions are given below −

i. "Man is Mortal", it returns truth value “TRUE”
ii. "12 + 9 = 3 – 2", it returns truth value “FALSE”

The following is not a Proposition –

"A is less than 2". It is because unless we give a specific value of A, we cannot say whether the statement is true or false.

Connectives

In propositional logic generally we use five connectives which are –

i. OR (v)
ii. AND (A)
iii. Negation/ NOT (¬)
iv. Implication / if-then (→)
v. If and only if (⇔).

OR (v) − The OR operation of two propositions A and B (written as *AvB*) is true if at least any of the propositional variable A or B is true. The truth table is as follows −

A	**B**	**A** v **B**
True	True	True
True	False	True
False	True	True
False	False	False

Table 3.1

AND (A**)** − The AND operation of two propositions A and B (written as *AAB*) is true if both the propositional variable A and B is true. The truth table is as follows −

A	**B**	**A** A **B**
True	True	True
True	False	False
False	True	False
False	False	False

Table 3.2

Negation (¬) − The negation of a proposition A (written as ¬*A*) is false when A is true and is true when A is false.The truth table is as follows −

A	**¬ A**
True	False

False	True

Table 3.3

Implication / if-then (→) − An implication $A \rightarrow B$ is the proposition "if A, then B". It is false if A is true and B is false. The rest cases are true. The truth table is as follows −

A	**B**	**A → B**
True	True	True
True	False	False
False	True	True
False	False	True

Table 3.4

If and only if (⇔) − $A \Leftrightarrow B$- isbiconditional logical connective which is true when p and q are same, i.e. both are false or both are true. The truth table is as follows −

A	**B**	**A ⇔ B**
True	True	True
True	False	False
False	True	False
False	False	True

Table 3.5

Tautologies - A Tautology is a formula which is always true for every value of its propositional variables.
Example − Prove $[(A \rightarrow B) \wedge A] \rightarrow B$ is a tautology
The truth table is as follows −

A	**B**	**A → B**	**(A → B)** ∧ **A**	**[(A → B)** ∧ **A] → B**
True	True	True	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	True

Table 3.6

As we can see every value of $[(A \rightarrow B) \wedge A] \rightarrow B$
is "True", it is a tautology.

Contradictions- A Contradiction is a formula which is always false for every value of its propositional variables.
Example − Prove $(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$ is a contradiction
The truth table is as follows −

A	**B**	**A ∨ B**	**¬ A**	**¬ B**	**(¬ A)** ∧ **(¬ B)**	**(A ∨ B)** ∧ **[(¬ A)** ∧ **(¬ B)]**
True	True	True	False	False	False	False
True	False	True	False	True	False	False
False	True	True	True	False	False	False
False	False	False	True	True	True	False

Table 3.7

As we can see every value of $(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$ is "False", it is a contradiction.

Contingency- A Contingency is a formula which has both some true and some false values for every value of its propositional variables.
Example − Prove $(A \vee B) \wedge (\neg A)$

The truth table is as follows −

A	**B**	**A ∨ B**	**¬ A**	**(A ∨ B) ∧ (¬ A)**
True	True	True	False	False
True	False	True	False	False
False	True	True	True	True
False	False	False	True	False

Table 3.8

As we can see every value of (*A*∨*B*)∧(¬*A*) has both "True" and "False", it is a contingency.

Laws of Algebra of Propositions:

Identity:

p ∨ p ≡ p p ∧ p ≡ p p → p ≡ T p ↔ p ≡ T
p ∨ T ≡ T p ∧ T ≡ p p → T ≡ T p ↔ T ≡ p
p ∨ F ≡ p p ∧ F ≡ F p → F ≡ ~p p ↔ F ≡ ~p
T → p ≡ p
F → p ≡ T

Commutative:

p ∨ q ≡ q ∨ p p ∧ q ≡ q ∧ p p → q ≠ q → p p ↔ q ≡ q ↔ p

Complement:

p ∨ ~p ≡ T p ∧ ~p ≡ F p → ~p ≡ ~p p ↔ ~p ≡ F
~p → p ≡ p

Double Negation:

~(~p) ≡ p

Associative:

p ∨ (q ∨ r) ≡ (p ∨ q) ∨ r
p ∧ (q ∧ r) ≡ (p ∧ q) ∧ r

Distributive:

p ∨ (q ∧ r) ≡ (p ∨ q) ∧ (p ∨ r)
p ∧ (q ∨ r) ≡ (p ∧ q) ∨ (p ∧ r)

Absorbtion:

p ∨ (p ∧ q) ≡ p
p ∧ (p ∨ q) ≡ p

De Morgan's:

~(p ∨ q) ≡ ~p ∧ ~q
~(p ∧ q) ≡ ~p ∨ ~q

Equivalence of Contrapositive:

p → q ≡ ~q → ~p

Others:

p → q ≡ ~p ∨ q
p ↔ q ≡ (p → q) ∧ (q → p)

<u>Propositional Equivalences</u>

Two statements X and Y are logically equivalent if any of the following two conditions hold −

i. The truth tables of each statement have the same truth values.
ii. The bi-conditional statement *X*⇔*Y*
iii. is a tautology.

Example − Prove ¬(*A*∨*B*)*and*[(¬*A*)∧(¬*B*)] are equivalent

Testing by 1st method (Matching truth table)

A	B	A ∨ B	¬ (A ∨ B)	¬ A	¬ B	[(¬ A) ∧ (¬ B)]
True	True	True	False	False	False	False
True	False	True	False	False	True	False
False	True	True	False	True	False	False
False	False	False	True	True	True	True

Table 3.9

Here, we can see the truth values of ¬(*A*∨*B*)*and*[(¬*A*)∧(¬*B*)]
are same, hence the statements are equivalent.
Testing by 2nd method (Bi-conditionality)

A	B	¬ (A ∨ B)	[(¬ A) ∧ (¬ B)]	[¬ (A ∨ B)] ⇔ [(¬ A) ∧ (¬ B)]
True	True	False	False	True
True	False	False	False	True
False	True	False	False	True
False	False	True	True	True

Table 3.10

As [¬(*A*∨*B*)]⇔[(¬*A*)∧(¬*B*)]
is a tautology, the statements are equivalent.

Inverse, Converse, and Contrapositive

1. Implication / if-then (→) is also called a conditional statement. It has two parts −
   i. Hypothesis, p
   ii. Conclusion, q

As mentioned earlier, it is denoted as $p \rightarrow q$

Example of Conditional Statement − "If you do your homework, you will not be punished." Here, "you do your homework" is the hypothesis, p, and "you will not be punished" is the conclusion, q.

1. **Inverse** − An inverse of the conditional statement is the negation of both the hypothesis and the conclusion. If the statement is "If p, then q", the inverse will be "If not p, then not q". Thus the inverse of $p \rightarrow q$is $\neg p \rightarrow \neg q$

Example − The inverse of "If you do your homework, you will not be punished" is "If you do not do your homework, you will be punished."

2. **Converse** − The converse of the conditional statement is computed by interchanging the hypothesis and the conclusion. If the statement is "If p, then q", the converse will be "If q, then p". The converse of $p \rightarrow q$is $q \rightarrow p$

Example − The converse of "If you do your homework, you will not be punished" is "If you will not be punished, you do not do your homework".

3. **Contrapositive** − Thecontrapositive of the conditional is computed by interchanging the hypothesis and the conclusion of the inverse statement. If the statement is "If p, then q", the contrapositive will be "If not q, then not p". The contrapositive of $p \rightarrow q$is $\neg q \rightarrow \neg p$

Example − The Contrapositive of " If you do your homework, you will not be punished" is "If you are not punished, then you do not do your homework".

<u>Duality Principle</u>

Duality principle states that for any true statement, the dual statement obtained by interchanging unions into intersections (and vice versa) and interchanging Universal set into Null set (and vice versa) is also true. If dual of any statement is the statement itself, it is said **self-dual** statement.

Example − The dual of $(A \cap B) \cup C$ is $(A \cup B) \cap C$

Predicate Logic- It deals with predicates, which are propositions containing variables.

Definition - A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.

The following are some examples of predicates –

i. Let E(x, y) denote "x = y"
ii. Let X(a, b, c) denote "a + b + c = 0"
iii. Let M(x, y) denote "x is married to y"

Normal Forms

We can convert any proposition in two normal forms −

i. Conjunctive normal form
ii. Disjunctive normal form

i. Conjunctive Normal Form - A compound statement is in conjunctive normal form if it is obtained by operating AND among variables (negation of variables included) connected with ORs. In terms of set operations, it is a compound statement obtained by Intersection among variables connected with Unions.

Example - (*A*v*B*)A(*A*v*C*)A(*B*v*C*v*D*)

ii. Disjunctive Normal Form- A compound statement is in conjunctive normal form if it is obtained by operating OR among variables (negation of variables included) connected with ANDs. In terms of set operations, it is a compound statement obtained by Union among variables connected with Intersections.

Example - (*AAB*)v(*AAC*)v(*BACAD*)

Well Formed Formula- Well Formed Formula (wff) is a predicate holding any of the following −

i. All propositional constants and propositional variables are wffs
ii. If x is a variable and Y is a wff, ∀*xY*and∃*xY*are also wff
iii. Truth value and false values are wffs Each atomic formula is a wff
iv. All connectives connecting wffs are wffs

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

1. **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 ∀.

∀*xP*(*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*)
where P(x) is the predicate which denotes x is mortal and the universe of discourse is all men.

2. **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 ∃.

∃*xP*(*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 ∃*xP*(*x*)
where P(x) is the predicate which denotes x is dishonest and the universe of discourse is some people.

3. **Nested Quantifiers -** If we use a quantifier that appears within the scope of another quantifier, it is called nested quantifier.

Example

i. ∀ *a*∃*bP*(*x*,*y*) where *P*(*a*,*b*) denotes *a*+*b*=0
ii. ∀ *a*∀*b*∀*cP*(*a*,*b*,*c*) where *P*(*a*,*b*) denotes *a*+(*b*+*c*)=(*a*+*b*)+*c*

Note − ∀*a*∃*bP*(*x*,*y*)≠∃*a*∀*bP*(*x*,*y*)

Rules of Inference- To deduce new statements from the statements whose truth that we already know, **Rules of Inference** are used.

- Mathematical logic is often used for logical proofs. Proofs are valid arguments that determine the truth

values of mathematical statements.

- An argument is a sequence of statements. The last statement is the conclusion and all its preceding statements are called premises (or hypothesis). The symbol "∴", (read therefore) is placed before the conclusion. A valid argument is one where the conclusion follows from the truth values of the premises.
- Rules of Inference provide the templates or guidelines for constructing valid arguments from the statements that we already have.

Table of Rules of Inference

Rule of Inference Name Rule of Inference Name

1. **Addition**

If P is a premise, we can use Addiction rule to derive *P*v*Q*

P*∴*P*v*Q

Example

Let P be the proposition, "He studies very hard" is true

Therefore − "Either he studies very hard Or he is a very bad student." Here Q is the proposition "he is a very bad student".

2. **Conjunction**

If P and Q are two premises, we can use Conjunction rule to derive *P*A*Q*

PQ*∴*P*A*Q

Example

Let P − "He studies very hard"

Let Q − "He is the best boy in the class"

Therefore − "He studies very hard and he is the best boy in the class"

3. **Simplification**

If *P*A*Q* is a premise, we can use Simplification rule to derive P.

P*A*Q*∴*P

Example

"He studies very hard and he is the best boy in the class", *P*A*Q*

Therefore − "He studies very hard"

4. **Modus Ponens**

If P and *P*→*Q* are two premises, we can use Modus Ponens to derive Q.

P*→*QP*∴*Q

Example

"If you have a password, then you can log on to facebook", *P*→*Q*

"You have a password", P Therefore − "You can log on to facebook"

5. **Modus Tollens**

If *P*→*Q*and ¬*Q* are two premises, we can use Modus Tollens to derive ¬*P*

P*→*Q*¬*Q*∴¬*P

Example

"If you have a password, then you can log on to facebook", *P*→*Q*

"You cannot log on to facebook", ¬*Q*

Therefore − "You do not have a password "

6. **Disjunctive Syllogism**

If ¬*P* and *P*v*Q* are two premises, we can use Disjunctive Syllogism to derive Q.

¬PP*v*Q*∴*Q

Example

"The ice cream is not vanilla flavored", ¬*P*

"The ice cream is either vanilla flavored or chocolate flavored", *P*v*Q*

Therefore − "The ice cream is chocolate flavored"

7. Hypothetical Syllogism

If $P \rightarrow Q$ and $Q \rightarrow R$ are two premises, we can use Hypothetical Syllogism to derive $P \rightarrow R$

$$P \rightarrow Q Q \rightarrow R \therefore P \rightarrow R$$

Example

"If it rains, I shall not go to school", $P \rightarrow Q$

"If I don't go to school, I won't need to do homework", $Q \rightarrow R$

Therefore − "If it rains, I won't need to do homework"

8. Constructive Dilemma

If $(P \rightarrow Q) \wedge (R \rightarrow S)$ and $P \vee R$ are two premises, we can use constructive dilemma to derive $Q \vee S$

$$(P \rightarrow Q) \wedge (R \rightarrow S) P \vee R \therefore Q \vee S$$

Example

"If it rains, I will take a leave", $(P \rightarrow Q)$

"If it is hot outside, I will go for a shower", $(R \rightarrow S)$

"Either it will rain or it is hot outside", $P \vee R$

Therefore − "I will take a leave or I will go for a shower"

9. Destructive Dilemma

If $(P \rightarrow Q) \wedge (R \rightarrow S)$ and $\neg Q \vee \neg S$ are two premises, we can use destructive dilemma to derive $P \vee R$

$$(P \rightarrow Q) \wedge (R \rightarrow S) \neg Q \vee \neg S \therefore P \vee R$$

Example

"If it rains, I will take a leave", $(P \rightarrow Q)$

"If it is hot outside, I will go for a shower", $(R \rightarrow S)$

"Either I will not take a leave or I will not go for a shower", $\neg Q \vee \neg S$

Therefore − "Either it rains or it is hot outside"

Finite state machine :a finite state machine (sometimes called a finite state automaton) is a computation model that can be implemented with hardware or software and can be used to simulate sequential logic and some computer programs. Finite state automata generate regular languages. Finite state machines can be used to model problems in many fields including mathematics, artificial intelligence, games, and linguistics.

Finite state machines as models of physical system equivalence machines:

Deterministic Finite-State Automata

A DFSA can be formally defined as A = (Q, Σ, ∂, q_0, F):

- Q, a finite set of states
- Σ, a finite alphabet of input symbols
- q0 ∈ Q, an initial start state
- F ⊆ Q, a set of final states
- ∂ (delta): Q x Σ → Q, a transition function

We can define ∂ on words, ∂_w, by using a recursive definition:

- ∂_w : Q x Σ* → Q — a function of (state, word) to a state
- $\partial_w(q,\varepsilon) = q$ — in state q, output state q if word is ε
- $\partial_w(q,xa) = \partial(\partial_w(q,x),a)$ otherwise, use ∂ for one step and recurse

For an automaton A, we can define the language of A:

- $L(A) = \{w \in \Sigma^* : \partial_w(q_0,w) \in F\}$
- L (A) is a subset of all words w of finite length over Σ, such that the transition function $\partial_w(q_0,w)$ produces a state in the set of final states (F).

- Intuitively, if we think of the automaton as a graph structure, then the words in L(A) represent the "paths" which end in a final state. If we concatenate the labels from the edges in each such path, we derive a string w ∈L(A).

1. States are shown as circles;
2. the start state is indicated by the bold incoming arrow.
3. The next state function and output functions are shown using directed arrows from one state to another. Each arrow is labeled with one element of I and one element of O.
4. If the machine is in some state s and the current input symbol is x, then we follow the arc labeled x/y from s to a new state and produce output y.

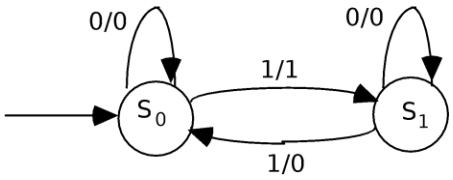


Figure 3.1

Finite State Machines as Language Recognizers

Input = {0,1}

acceptingstate : A state is said to be an accepting state if its output is 1.

rejectingstate : A state is said to be an rejecting state if its output is 0.

An input sequence is said to be accepted by the finite state machine if it leads the machine from the initial state to an accepting state. On the other hand, an input sequence is said to be rejected by the finite state machine if it leads the machine from the initial state to an rejecting state.

finite state language

A language is said to be a finite state language if there is a finite state machine that accepts exactly all sentences in the language.

Theorem Let L be a finite state language accepted by a finite state machine with N states. For any sequence α whose length is N or larger in the language, α can be written as uvw such that v is nonempty and uv^iw is also in the language for i≧0, where vi denotes the concatenation of i copies of the sequence v. (In other words, uw, uvw, uvvw, uvvvw, … are all in the language.)

Proof :

Letα= a1a2a3…aN, without loss of generality.

Let s_{j0}, s_{j1}, s_{j2}, …,s_{jN} denote the states the machine visits, where s_{j0} is the initial state and s_{jN} is an accepting state.

Among the N+1 states s_{j0}, s_{j1}, s_{j2}, …,s_{jN} there are two of them that are the same. Suppose that is state s_k, we realize that the sequences uw, uvw, uvvw, uvvvw, …, uv^lw, … will all lead the machine from the initial state s_{j0} to the accepting state s_{jN}.

Question :Design a DFA such that: L = {$a^nb^mc^l$ | n,m,l ≥ 1} Given: Input alphabet, Σ={a, b, c} Language L = {abc, aabc, abbc, abcc, ...}

Clearly the language is infinite because there is infinite number of strings.

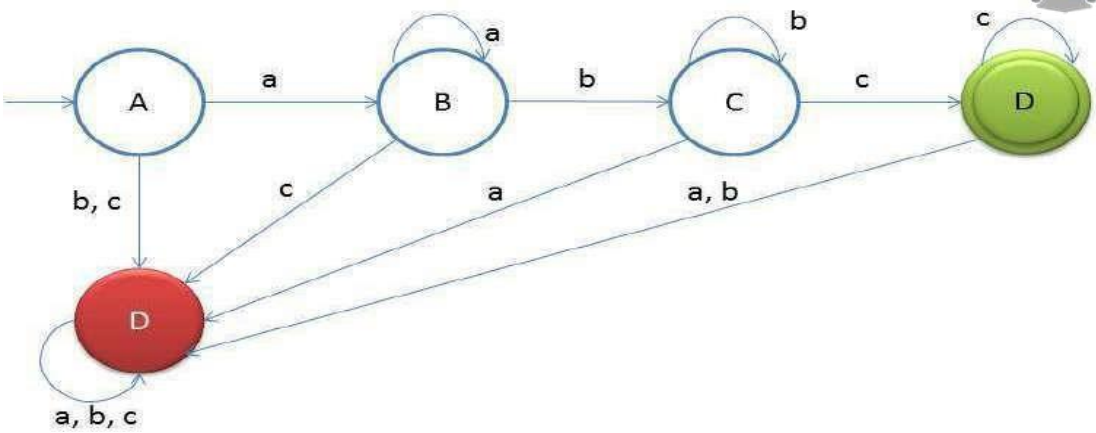


Figure 3.2

The idea behind this approach is very simple, follow the steps below and you will understand.

1. Accept the smallest string 'abc' in the DFA that is four states will be made and state D will be made the final state
2. Now what if input 'b'/'c' comes on state A then it should be rejected as 'b'/'c' cannot come before 'a'
3. At state B if 'a' comes then we can loop it as no given DFA property will be violated
4. At state B if 'c' comes then it should be rejected as 'c' cannot come before 'b'
5. At state C if 'b' comes then we will loop it
6. At state C if 'a' comes then it should be rejected as 'a' cannot come after 'b'
7. At state C if 'c' comes then it should flow to state D
8. At state D if 'c' comes then we will loop it
9. At state D if 'a'/'b' comes then it should be rejected as 'a'/'b' cannot come after 'c'
10. State E will be Dead state and it will be looped for 'a', 'b' and 'c'

Testing

1. Lets take one input string aaabbbcc
2. Scan string from left to right
3. First input is a, so from state A we will go to state B
4. Second input is a, so from state B we will go to state B itself
5. Third input is a, so from state B we will go to state B itself
6. Fourth input is b, so from state B we will go to state C
7. Fifth input is b, so from state C we will go to state C itself
8. Sixth input is b, so from state C we will go to state C itself
9. Seventh input is c, so from state C we will go to state D
10. Eight input is c, so from state C we will go to state C itself(final state)

Subject Notes
Discrete Structures

UNIT-4

Graph

Definition − A graph (denoted as *G*= (*V*, *E*)) consists of a non-empty set of vertices or nodes V and a set of edges E.

Example − Let us consider, a Graph is *G*=(*V*,*E*)where *V*={*a*, *b*, *c*, *d*} and *E*={{*a*,*b*},{*a*,*c*},{*b*,*c*},{*c*,*d*}}

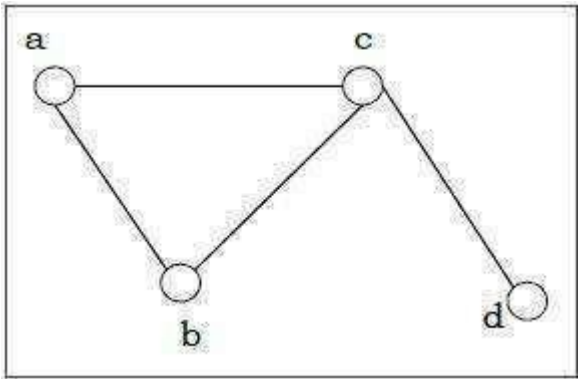


Figure 4.1 example of graph

Degree of a Vertex − the degree of a vertex V of a graph G (denoted by deg (V)) is the number of edges incident with the vertex V.

Vertex	Degree	Even / Odd
a	2	even
b	2	even
c	3	odd
d	1	odd

Even and Odd Vertex − If the degree of a vertex is even, the vertex is called an even vertex and, if the degree of a vertex is odd, the vertex is called an odd vertex

Degree of a Vertex − the degree of a graph is the largest vertex degree of that graph. For the above graph the degree of the graph is 3.

The Handshaking Lemma − in a graph, the sum of all the degrees of all the vertices is equal to twice the number of edges.

Types of Graphs

There are different types of graph

1. **Null Graph -** A null graph has no edges. The null graph of *n* vertices is denoted by *Nn*

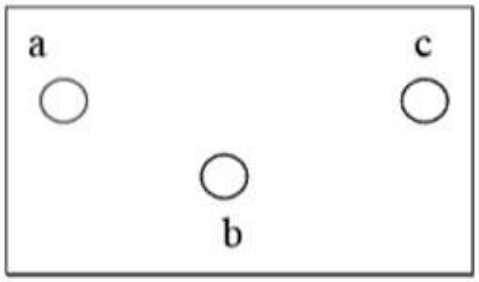


Figure 4.2 Null Graph

2. **Simple Graph** -A graph is called simple graph/strict graph if the graph is undirected and does not contain any loops or multiple edges.

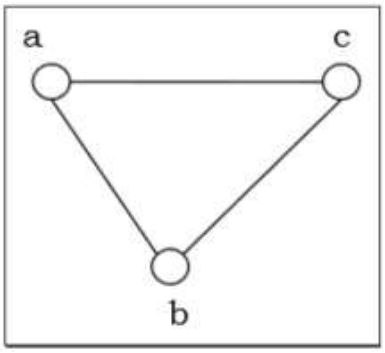


Figure 4.3 Simple Graph

3. **Directed and Undirected Graph -** A graph G=(V,E) is called a directed graph if the edge set is made of ordered vertex pair and a graph is called undirected if the edge set is made of unordered vertex pair.

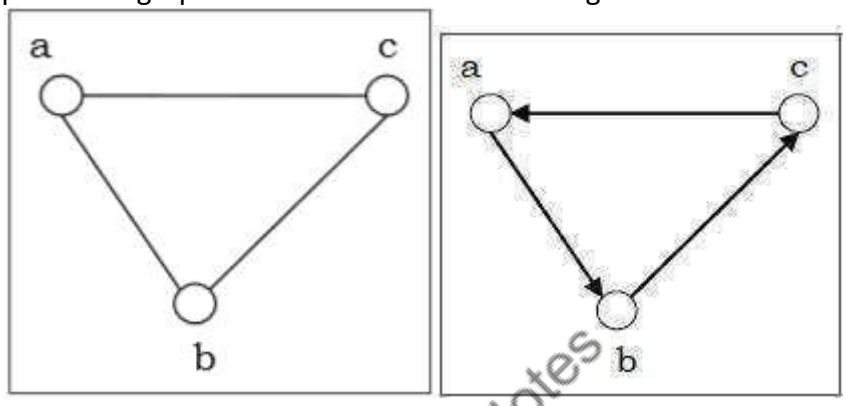


Figure 4.4 Directed and Undirected Graphs

4. **Connected and Disconnected Graph -** A graph is connected if any two vertices of the graph are connected by a path; while a graph is disconnected if at least two vertices of the graph are not connected by a path. If a graph G is disconnected, then every maximal connected subgraph of G is called a connected component of the graph G

.

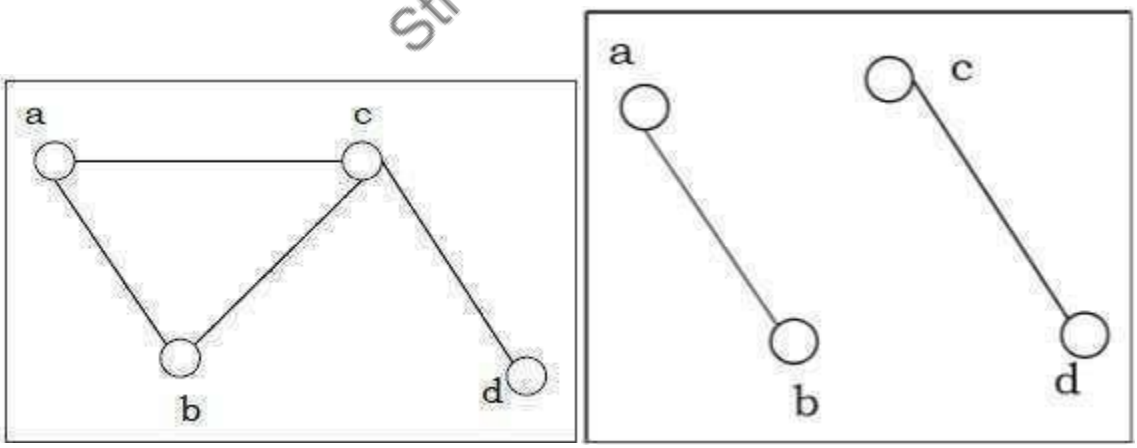


Figure 4.5 Connected and Disconnected Graph

5. **Regular Graph -** A graph is regular if all the vertices of the graph have the same degree. In a regular graph G of degree r, the degree of each vertex of G is r.

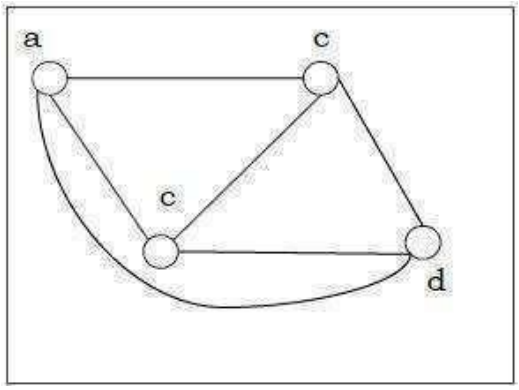


Figure 4.6 Regular Graph

6. **Complete Graph-** A graph is called complete graph if every two vertices pair are joined by exactly one edge. The complete graph with n vertices is denoted by Kn

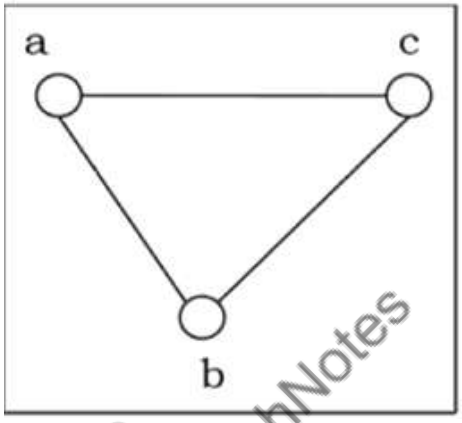


Figure 4.7 Complete Graph

7. **Cycle Graph-** If a graph consists of a single cycle, it is called cycle graph. The cycle graph with n vertices is denoted by Cn

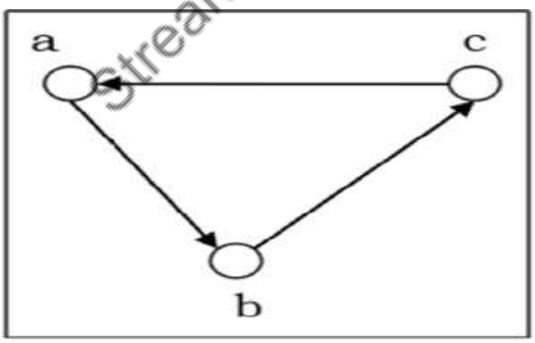


Figure 4.8 Cycle Graph

8. **Bipartite Graph -** If the vertex-set of a graph G can be split into two disjoint sets, V1 and V2, in such a way that each edge in the graph joins a vertex in V1 to a vertex in V2, and there are no edges in G that connect two vertices in V1 or two vertices in V2, then the graph G is called a bipartite graph.

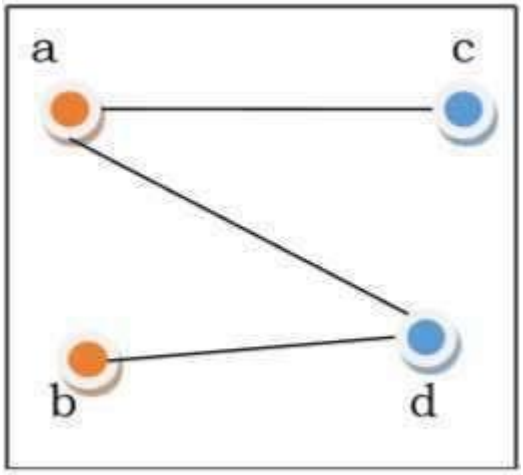


Figure 4.9 Bipartite Graph

9. **Complete Bipartite Graph-** A complete bipartite graph is a bipartite graph in which each vertex in the first set is joined to every single vertex in the second set. The complete bipartite graph is denoted by Kx,ywhere the graph G contains x vertices in the first set and y vertices in the second set.

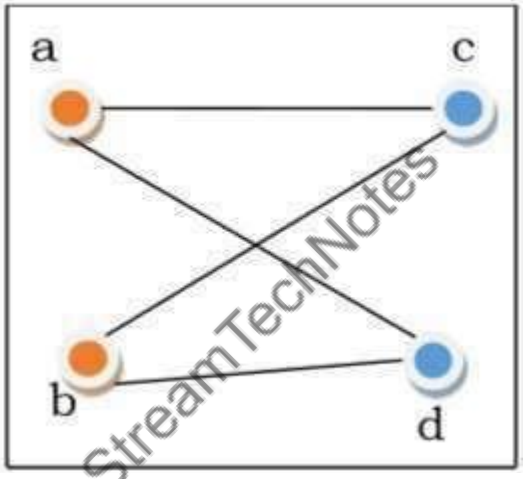


Figure 4.10 Complete Bipartite Graphs

10. **Planar vs. Non-planar graph**

Planar graph – A graph G is called a **planar** graph if it can be drawn in a plane without any edges crossed. If we draw graph in the plane without edge crossing, it is called embedding the graph in the plane.

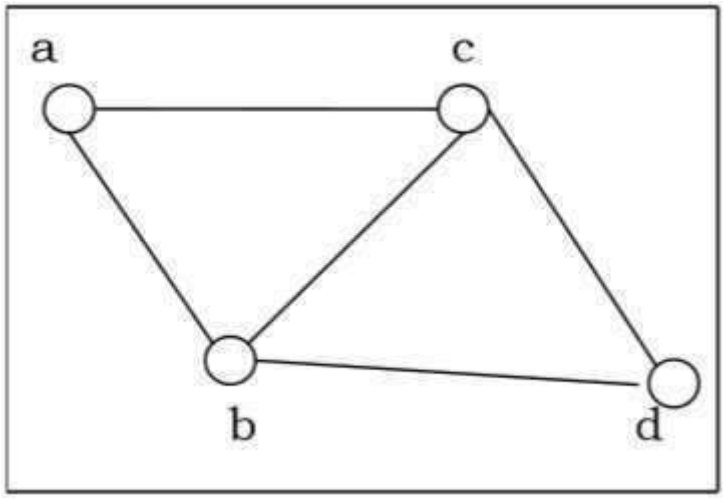


Figure 4.11 Planar graph

Non-planar graph – A graph is non-planar if it cannot be drawn in a plane without graph edges crossing.

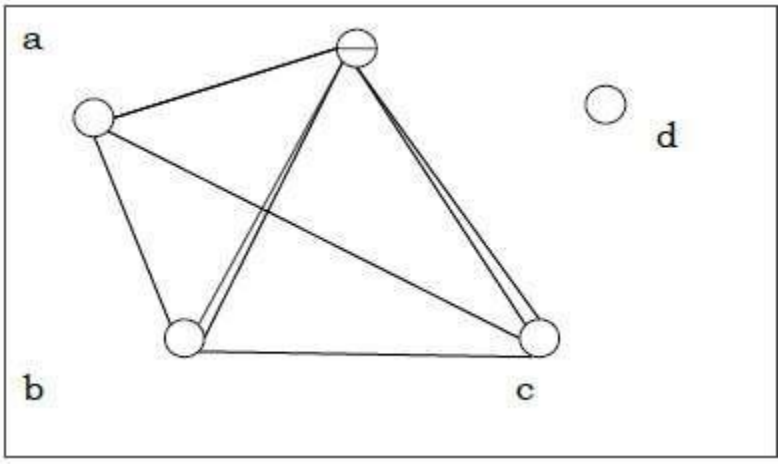


Figure 4.12 Non-planar graph

11. **Multi-Graph-** If in a graph multiple edges between the same set of vertices are allowed, it is called Multigraph. In other words, it is a graph having at least one loop or multiple edges.

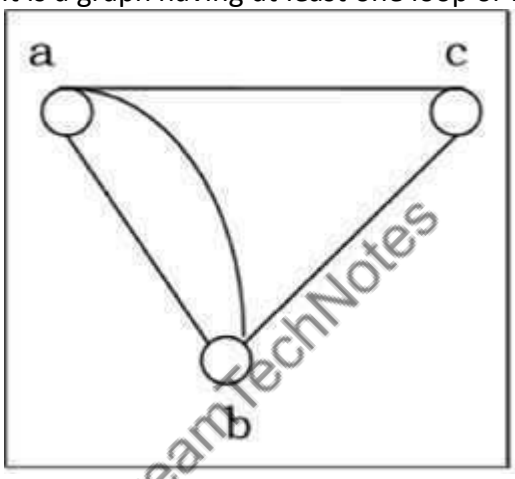


Figure 4.13 Multi-Graph

12. **Weighted Graph:** weighted graph is a graph in which each branch is given a numerical weight. A weighted graph is therefore a special type of labeled graph in which the labels are numbers (which are usually taken to be positive).

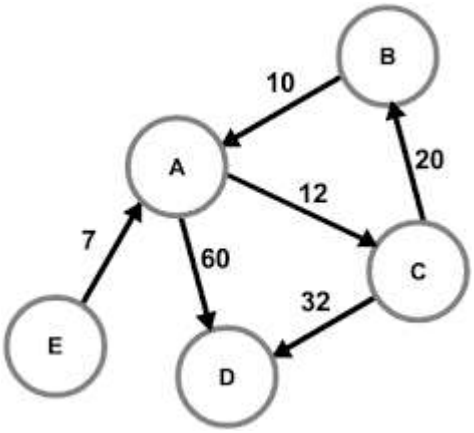


Figure 4.14 Weighted Graphs

13. **Isomorphic graph** - If two graphs G and H contain the same number of vertices connected in the same way, they are called isomorphic graphs (denoted by G≅H).
It is easier to check non-isomorphism than isomorphism. If any of these following conditions occurs, then two graphs are non-isomorphic −

1) The number of connected components are different

2) Vertex-set cardinalities are different
3) Edge-set cardinalities are different
4) Degree sequences are different

Example

The following graphs are isomorphic −

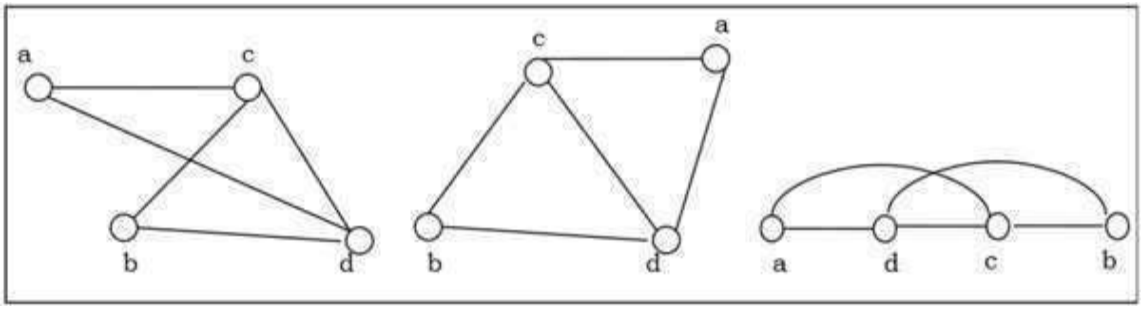


Figure 4.15 Isomorphic graph

Representation of Graphs

There are mainly two ways to represent a graph −

1. Adjacency Matrix
2. Adjacency List

1. **Adjacency Matrix -** AnAdjacency Matrix *A*[*V*][*V*] is a 2D array of size *V*×*V* where *V* is the number of vertices in a undirected graph. If there is an edge between *Vx* to *Vy* then the value of *A*[*Vx*][*Vy*]=1 and *A*[*Vy*][*Vx*]=1, otherwise the value will be zero. And for a directed graph, if there is an edge between *Vx* to *Vy*, then the value of *A*[*Vx*][*Vy*]=1 , otherwise the value will be zero.

Adjacency Matrix of an Undirected Graph

Let us consider the following undirected graph and construct the adjacency matrix −

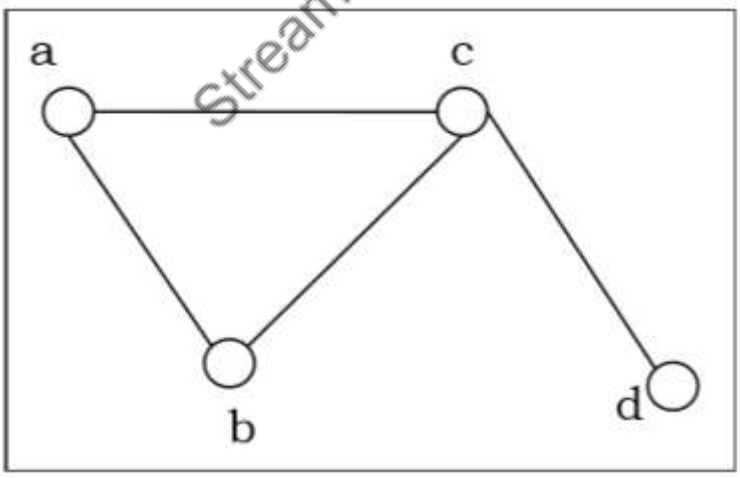


Figure 4.16 Adjacency Matrix of an Undirected Graph

Adjacency matrix of the above undirected graph will be −

	a	b	c	d
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

Adjacency Matrix of a Directed Graph

Let us consider the following directed graph and construct its adjacency matrix −

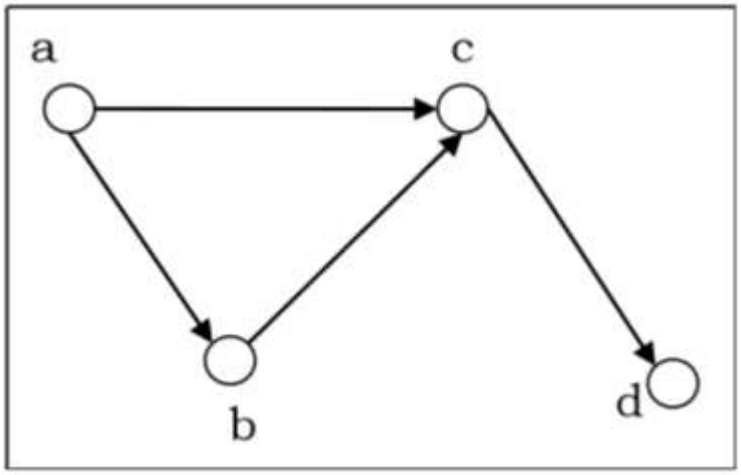


Figure 4.17 Adjacency Matrix of a Directed Graph

Adjacency matrix of the above directed graph will be −

	a	**b**	**c**	**d**
a	0	1	1	0
b	0	0	1	0
c	0	0	0	1
d	0	0	0	0

2. **Adjacency List - In** adjacency list, an array (*A*[*V*]) of linked lists is used to represent the graph G with *V* number of vertices. An entry *A*[*Vx*] represents the linked list of vertices adjacent to the *Vx*−*th*vertex. The adjacency list of the undirected graph is as shown in the figure below −

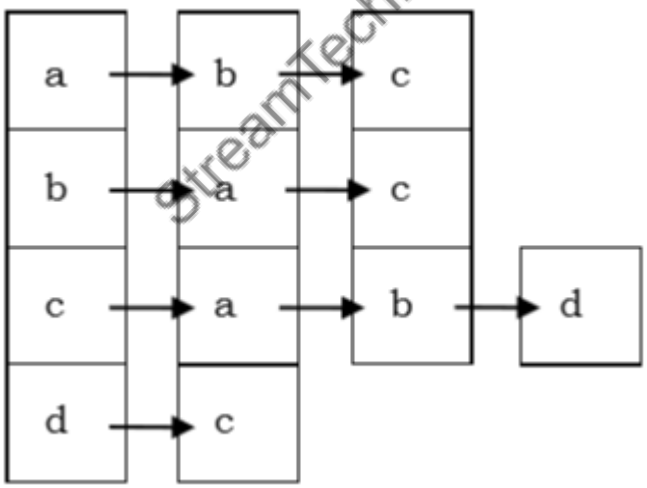


Figure 4.18 Adjacency List

Cycles and connectivity

Definitions:

- **Walk**: finite sequence of edges in which any two consecutive edges are adjacent or identical. (Initial vertex, Final vertex, length)
- **Trail**: walk with all distinct edges
- **Path**: A path is a sequence of vertices with the property that each vertex in the sequence is adjacent to the vertex next to it. A path that does not repeat vertices is called a simple path.
- **Circuit**: A circuit is path that begins and ends at the same vertex.
- **Cycle**: A circuit that doesn't repeat vertices is called a cycle.

Shortest path in weighted graph: Given a graph where edges are labeled with weights (or distances) and a source vertex, what is the shortest path between the source and some other vertex? Problems requiring us to answer such queries are broadly known as shortest-paths problems. Shortest-paths problem come in several flavors. For example, the single-source shortest path problem requires finding the shortest paths between a given source and all other vertices; the single-pair shortest path problem requires finding the shortest path between given a source and a given destination vertex; the all-pairs shortest path problem requires finding the shortest paths between all pairs of vertices.

Shortest Paths: Problem Statement

- Given a weighted graph and two vertices u and v, we want to find a path of minimum total weight between u and v
- Length (or distance) of a path is the sum of the weights of its edges Length (or distance) of a path is the sum of the weights of its edges

Applications

- Internet packet routing
- Flight reservations
- Driving directions

Assumptions

1) ƒ Graph is simple
    a. ƒ No parallel edges and no self-loops
2) ƒ Graph is connected
    a. ƒ If not, run the algorithm for each connected component
3) ƒ Graph is undirected Graph is undirected
    a. ƒ It is simple to extend to directed case
4) ƒ No negative weight edges
    a. ƒ There is an algorithm to compute shortest paths in a graph with negative edges
    b. ƒ It has higher time complexity
    c. ƒ Does not work if there is a negative cost cycle Does not work if there is a negative cost cycle
    d. ƒ Makes no sense to compute shortest paths in the presence of negative cycles
        i. ƒ in a graph with a negative cycle, shortest path has cost negative infinity

Dijkstra's algorithm

Steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
3) While sptSet doesn't include all vertices
    a) Pick a vertex u which is not there in sptSetand has minimum distance value.
    b) Include u to sptSet.
    c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Let us understand with the following example:

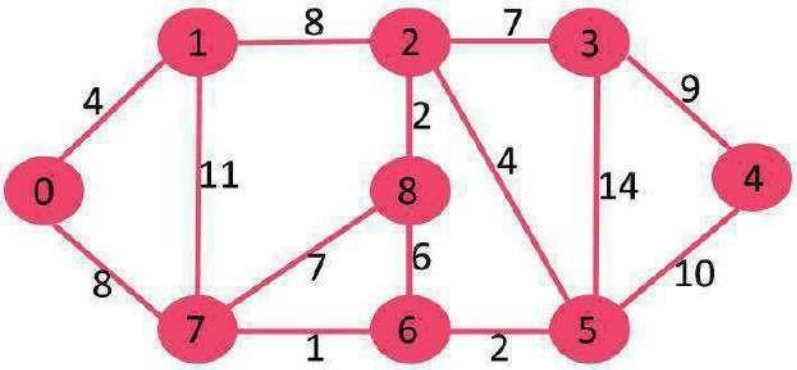


Figure 4.19 example of shortest Path

- The set sptSetis initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite.
- Now pick the vertex with minimum distance value. The vertex 0 is picked, include it in sptSet. So sptSet becomes {0}. After including 0 to sptSet, update distance values of its adjacent vertices.
- Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8.
- Following subgraph shows vertices and their distance values, only the vertices with finite distance values are shown. The vertices included in SPT are shown in green color.

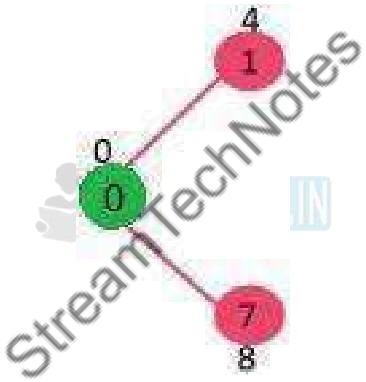


Figure 4.19 (a)

- Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1.
- The distance value of vertex 2 becomes 12.

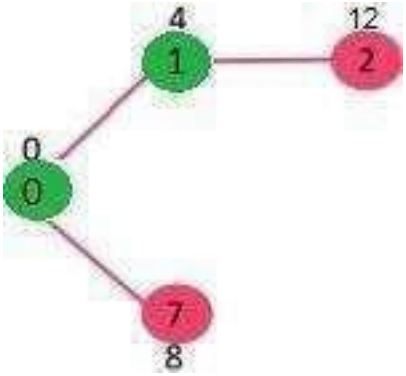


Figure 4.19 (b)

- Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (15 and 9 respectively).

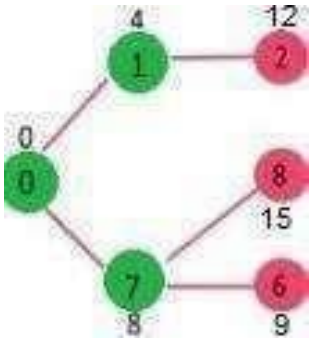


Figure 4.19 (c)

- Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.

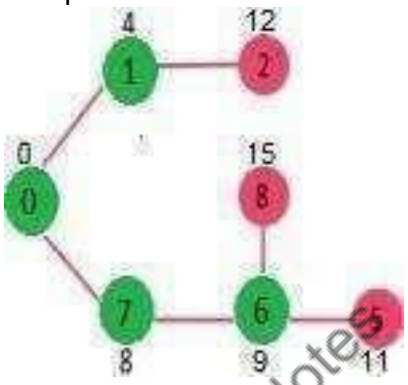


Figure 4.19 (d)

- We repeat the above steps until sptSet doesn't include all vertices of given graph. Finally, we get the following Shortest Path Tree (SPT).

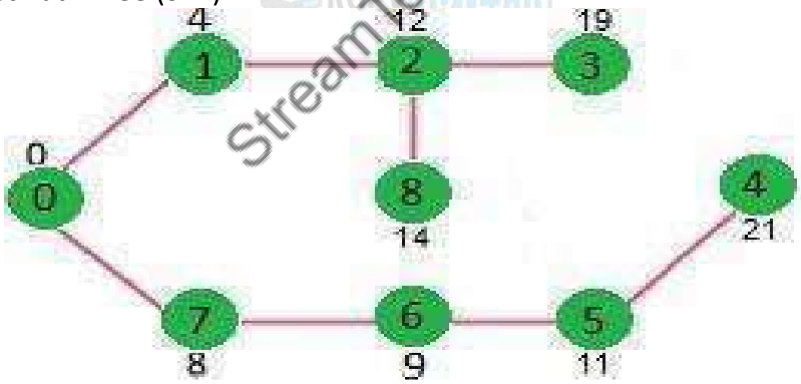


Figure 4.19 (e)

Introduction to Eulerian paths and Circuits

Euler Graphs - A connected graph *G* is called an Euler graph, if there is a closed trail which includes every edge of the graph *G* . An Euler path is a path that uses every edge of a graph exactly once. An Euler path starts and ends at different vertices. An Euler circuit is a circuit that uses every edge of a graph exactly once. An Euler circuit always starts and ends at the same vertex. A connected graph *G* is an Euler graph if and only if all vertices of *G* are of even degree, and a connected graph *G* is Eulerian if and only if its edge set can be decomposed into cycles.

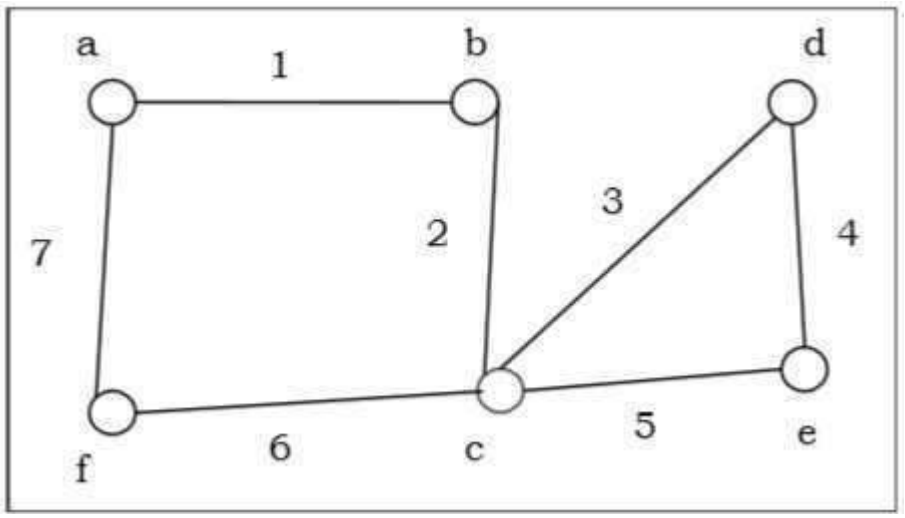


Figure 4.20 Euler Graph

The above graph is an Euler graph as "*a*1*b*2*c*3*d*4*e*5*c*6*f*7*g*"covers all the edges of the graph.

Eulerian path and circuit for undirected graph

Eulerian Path is a path in graph that visits every edge exactly once. Eulerian Circuit is an Eulerian Path which starts and ends on the same vertex.

How to find whether a given graph is Eulerian or not?

- The problem is same as following question. "Is it possible to draw a given graph without lifting pencil from the paper and without tracing any of the edges more than once".
- A graph is called Eulerian if it has an Eulerian Cycle and called Semi-Eulerian if it has an Eulerian Path.
- The problem is NP complete problem for a general graph. Fortunately, we can find whether a given graph has a Eulerian Path or not in polynomial time. In fact, we can find it in O(V+E) time.
- Following are some interesting properties of undirected graphs with an Eulerian path and cycle. We can use these properties to find whether a graph is Eulerian or not.

Eulerian Cycle an undirected graph has Eulerian cycle if following two conditions are true.

1) All vertices with non-zero degree are connected. We don't care about vertices with zero degree because they don't belong to Eulerian Cycle or Path (we only consider all edges).
2) All vertices have even degree.

Eulerian Path An undirected graph has Eulerian Path if following two conditions are true.

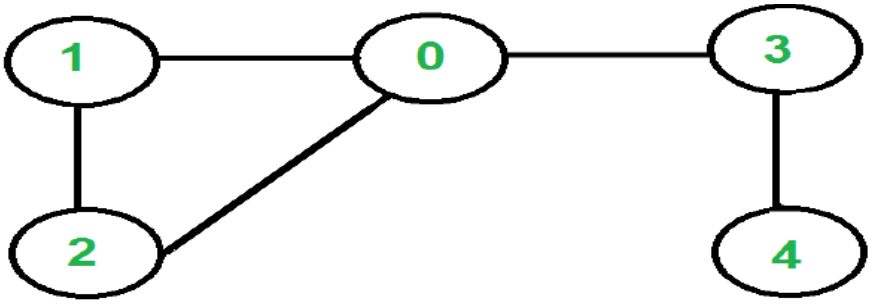
1) Same as condition
    a. For Eulerian Cycle
2) If zero or two vertices have odd degree and all other vertices have even degree.

Note: that only one vertex with odd degree is not possible in an undirected graph (sum of all degrees is always even in an undirected graph)

Note: that a graph with no edges is considered Eulerian because there are no edges to traverse.

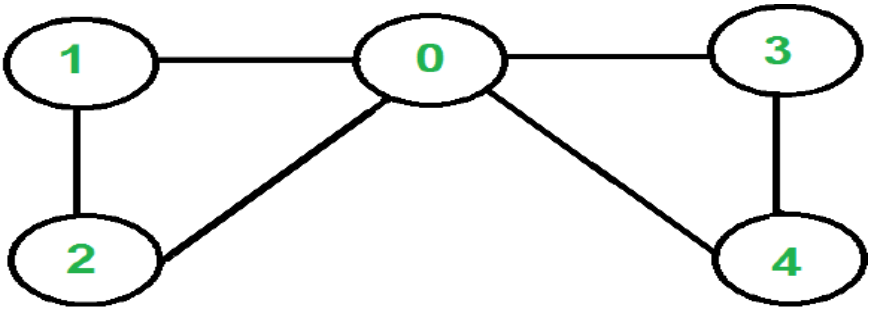
How does this work?

In Eulerian path, each time we visit a vertex v, we walk through two unvisited edges with one end point as v. Therefore, all middle vertices in Eulerian Path must have even degree. For Eulerian Cycle, any vertex can be middle vertex; therefore all vertices must have even degree.



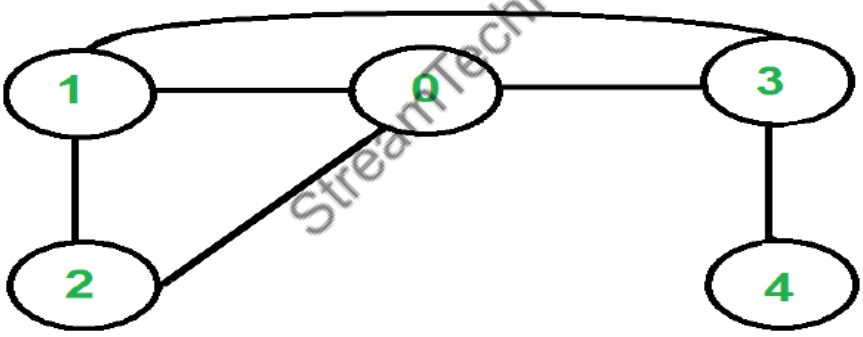
The graph has Eulerian Paths, for example "4 3 0 1 2 0", but no Eulerian Cycle. Note that there are two vertices with odd degree (4 and 0)

Figure 4.21 (a) example of Eulerian path



The graph has Eulerian Cycles, for example "2 1 0 3 4 0 2" Note that all vertices have even degree

Figure 4.21 (b) example of Eulerian cycle



The graph is not Eulerian. Note that there are four vertices with odd degree (0, 1, 3 and 4)

Figure 4.21 (c) examples with no Eulerian path

Hamiltonian Graphs - A connected graph *G* is called Hamiltonian graph if there is a cycle which includes every vertex of *G* and the cycle is called Hamiltonian cycle. Hamiltonian walk in graph *G* is a walk that passes through each vertex exactly once. If *G* is a simple graph with n vertices, where $n \geq 3$ If $deg(v) \geq n2$ for each vertex *v*, then the graph *G* is Hamiltonian graph. This is called Dirac's Theorem. If *G* is a simple graph with *n* vertices, where $n \geq 2$ if $deg(x)+deg(y) \geq n$ for each pair of non-adjacent vertices x and y, then the graph *G* is Hamiltonian graph. This is called Ore's theorem.

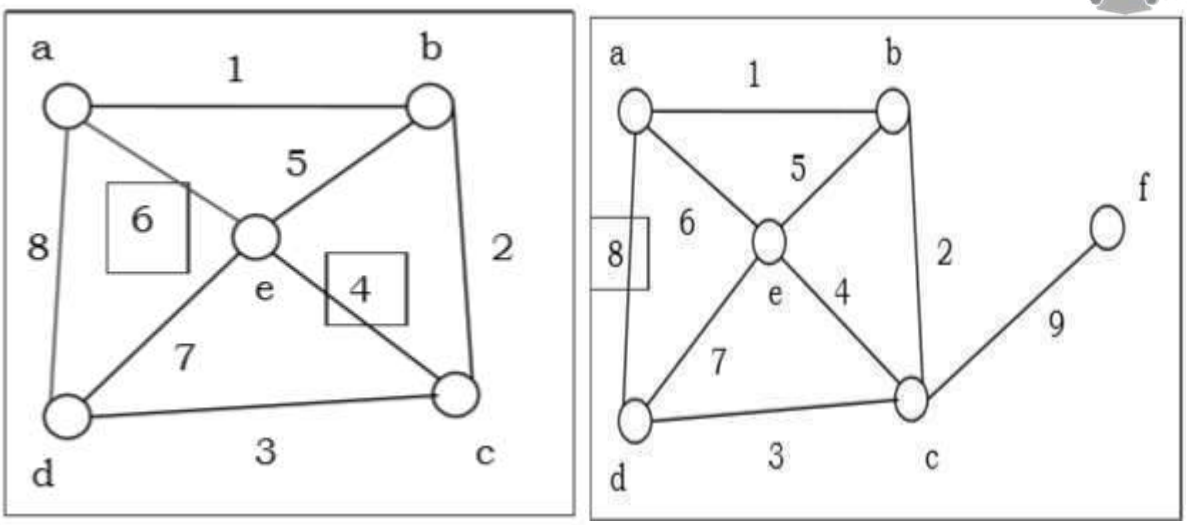


Figure 4.22 Hamiltonian Graphs

Hamiltonian Circuits A Hamiltonian circuit is a circuit that visits each of the vertices once and only once and ends on the same vertex as it began. For example, in this network the Hamiltonian circuit is marked in red. As it is a circuit, we cannot have repeated edges. We do not need to use all the edges, just visit each vertex once.

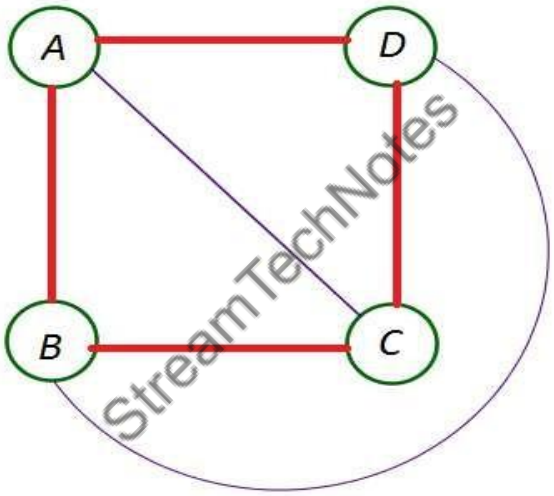


Figure 4.23 Hamiltonian Circuits

Other Hamiltonian circuits here include ABDCA, ABDCA, DBACD, DBACD and many others. If a graph has a Hamiltonian circuit then it automatically has a Hamiltonian path. (By just dropping off the last vertex in the circuit we create a path, for example, ABDC and DBAC from above)

Hamiltonian path: A Hamiltonian path is a path that visits each of the vertices once and only once but can begin and end on different vertices. For example, the Hamiltonian path in the network here could be ABCDE, ABCDE.

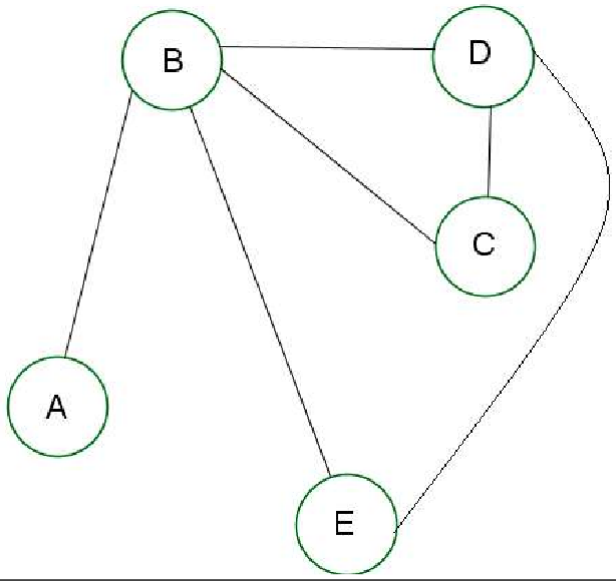


Figure 4.23 example of Hamiltonian path

Unlike with the Euler paths and Euler circuits, there is no single rule or theorem to help us identify if a Hamiltonian path or Hamiltonian circuit exists. The only thing we can do is look for them.
The following network has both Hamiltonian path and Hamiltonian circuit - can you find them both?

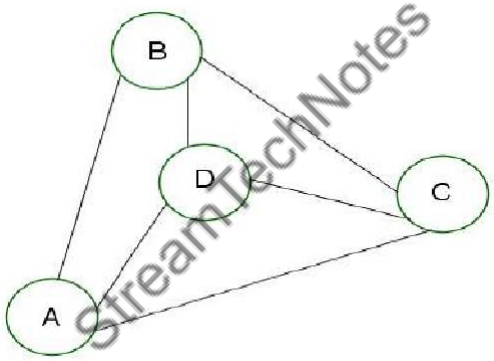


Figure 4.23 examples of both Hamiltonian path & Hamiltonian Circuits

Remember, the Hamiltonian path is a path where we visit each of the vertices only once. There are many Hamiltonian Paths here,

Graph Coloring- Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem.

Chromatic number: The chromatic number of a graph G is the smallest number of colors needed to color the vertices of G so that no two adjacent vertices share the same color i.e., the smallest value of k possible to obtain a k-coloring.

Method to Color a Graph

The steps required to color a graph G with n number of vertices are as follows −

Step 1 − Arrange the vertices of the graph in some order.

Step 2 − Choose the first vertex and color it with the first color.

Step 3 − Choose the next vertex and color it with the lowest numbered color that has not been colored on any vertices adjacent to it. If all the adjacent vertices are colored with this color, assign a new color to it. Repeat this step until all the vertices are colored.

Example

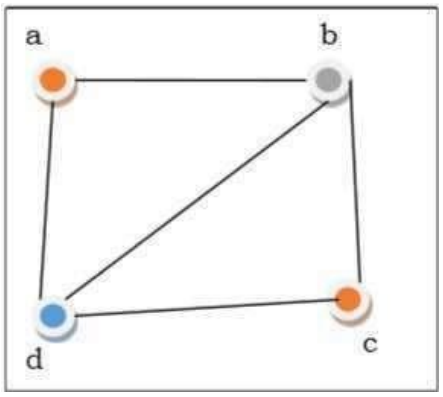


Figure 4.24 Graph coloring

In the above figure, at first vertex *a* is colored red. As the adjacent vertices of vertex a are again adjacent, vertex *b* and vertex *d* are colored with different color, green and blue respectively. Then vertex *c* is colored as red as no adjacent vertex of *c* is colored red. Hence, we could color the graph by 3 colors. Hence, the chromatic number of the graph is 3.

Applications of Graph Coloring- Some applications of graph coloring include

1. Register Allocation
2. Map Coloring
3. Bipartite Graph Checking
4. Mobile Radio Frequency Assignment
5. Making time table, etc.

Isomorphism of Graphs - If two graphs G and H contain the same number of vertices connected in the same way, they are called isomorphic graphs (denoted by $G \cong H$).

It is easier to check non-isomorphism than isomorphism. If any of these following conditions occurs, then two graphs are non-isomorphic −

1. The number of connected components are different
2. Vertex-set cardinalities are different
3. Edge-set cardinalities are different
4. Degree sequences are different

Example

The following graphs are isomorphic −

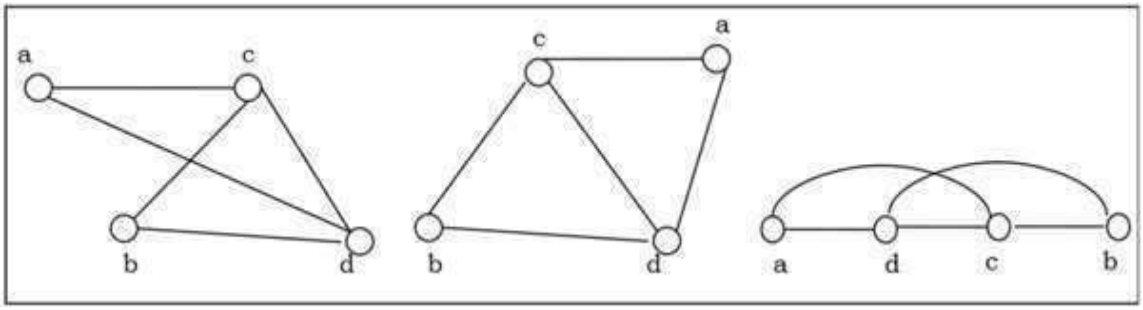


Figure 4.25 Isomorphism of Graph

Homomorphism - A homomorphism from a graph *G* to a graph *H* is a mapping (May not be a bijective mapping)$h:G\rightarrow H$ such that – $(x,y)\in E(G)\rightarrow(h(x),h(y))\in E(H)$. It maps adjacent vertices of graph *G* to the adjacent vertices of the graph *H.*

Properties of Homomorphisms

1. A homomorphism is an isomorphism if it is a bijective mapping.
2. Homomorphism always preserves edges and connectedness of a graph.
3. The compositions of homomorphisms are also homomorphisms.
4. To find out if there exists any homomorphic graph of another graph is aNPcomplete problem.

Graph Traversal

Graph traversal is the problem of visiting all the vertices of a graph in some systematic order. There are mainly two ways to traverse a graph.

1. **Breadth First Search**
2. **Depth First Search**

1. **Breadth First Search** - Breadth First Search (BFS) starts at starting level-0 vertex X of the graph G. Then we visit all the vertices that are the neighbors of X . After visiting, we mark the vertices as "visited," and place them into level-1. Then we start from the level-1 vertices and apply the same method on every level-1 vertex and so on. The BFS traversal terminates when every vertex of the graph has been visited.

BFS Algorithm

The concept is to visit all the neighbor vertices before visiting other neighbor vertices of neighbor vertices.

1. Initialize status of all nodes as “Ready”.
2. Put source vertex in a queue and change its status to “Waiting”.
3. Repeat the following two steps until queue is empty –
4. Remove the first vertex from the queue and mark it as “Visited”.
5. Add to the rear of queue all neighbors of the removed vertex whose status is “Ready”. Mark their status as “Waiting”.

Problem

Let us take a graph (Source vertex is ‘a’) and apply the BFS algorithm to find out the traversal order.

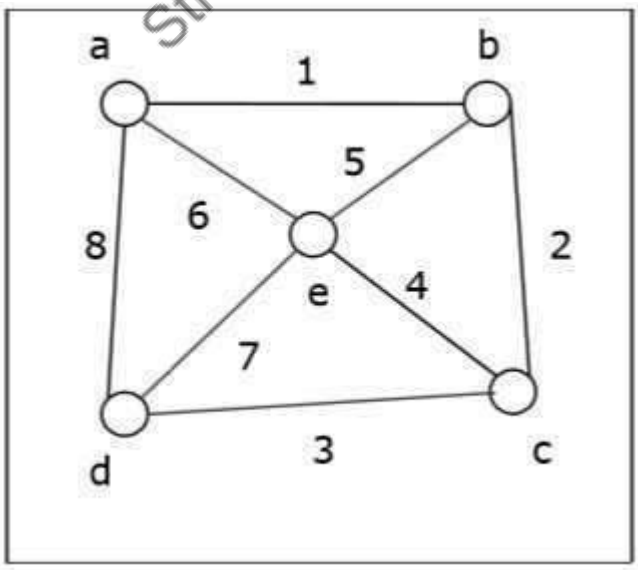


Figure 4.26 examples of BFS

Solution –

- Initialize status of all vertices to “‘eady”.
- Put a in queue and change its status to “Waiting”.
- ‘emove a from queue, mark it as “Visited”.
- Add a’s neighbors in “‘eady” state b, d and e to end of queue and mark them as “Waiting”.

- Remove b from queue, mark it as “Visited”, put its “‘eady” neighbor c at end of queue and mark c as “Waiting”.
- ‘emove d from queue and mark it as “Visited”. It has no neighbor in “‘eady” state.
- ‘emove e from queue and mark it as “Visited”. It has no neighbor in “‘eady” state.
- Remove c from queue and mark it as “Visited”. It has no neighbor in “‘eady” state.
- Queue is empty so stop.

So the traversal order is −

$a \rightarrow b \rightarrow d \rightarrow e \rightarrow c$

The alternate orders of traversal are −

$a \rightarrow b \rightarrow e \rightarrow d \rightarrow c$

Or, $a \rightarrow d \rightarrow b \rightarrow e \rightarrow c$

Or, $a \rightarrow e \rightarrow b \rightarrow d \rightarrow c$

Or, $a \rightarrow b \rightarrow e \rightarrow d \rightarrow c$

Or, $a \rightarrow d \rightarrow e \rightarrow b \rightarrow c$

Application of BFS

1. Finding the shortest path
2. Minimum spanning tree for un-weighted graph
3. GPS navigation system
4. Detecting cycles in an undirected graph
6. Finding all nodes within one connected component

Complexity Analysis - Let $G(V, E)$ be a graph with $|V|$ number of vertices and $|E|$ number of edges. If breadth first search algorithm visits every vertex in the graph and checks every edge, then its time complexity would be −

$$O(|V|+|E|).O(|E|)$$

It may vary between $O(1)$ and $O(|V2|)$

1. **Depth First Search -** Depth First Search (DFS) algorithm starts from a vertex *v* , then it traverses to its adjacent vertex (say x) that has not been visited before and mark as "visited" and goes on with the adjacent vertex of *x* and so on. If at any vertex, it encounters that all the adjacent vertices are visited, then it backtracks until it finds the first vertex having an adjacent vertex that has not been traversed before. Then, it traverses that vertex, continues with its adjacent vertices until it traverses all visited vertices and has to backtrack again. In this way, it will traverse all the vertices reachable from the initial vertex *v.*

DFS Algorithm

The concept is to visit all the neighbor vertices of a neighbor vertex before visiting the other neighbor vertices.

- Initialize status of all nodes as “‘eady”
- Put source vertex in a stack and change its status to “Waiting”
- Repeat the following two steps until stack is empty –
- Pop the top vertex from the stack and mark it as “Visited”
- Push onto the top of the stack all neighbors of the removed vertex whose status is “‘eady”. Mark their status as “Waiting”.

Problem

Let us take a graph (Source vertex is ‘a’) and apply the DFS algorithm to find out the traversal order.

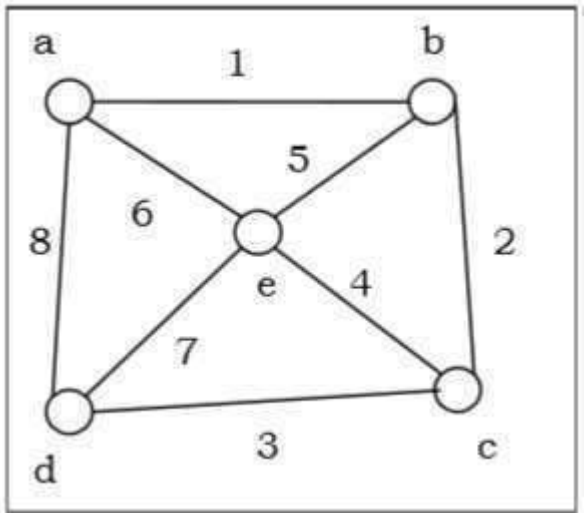


Figure 4.27 examples of DFS

Solution

- Initialize status of all vertices to "'eady".
- Push a in stack and change its status to "Waiting".
- Pop a and mark it as "Visited".
- Push a's neighbors in "'eady" state e, d and b to top of stack and mark them as "Waiting".
- Pop b from stack, mark it as "Visited", push its "'eady" neighbor c onto stack.
- Pop c from stack and mark it as "Visited". It has no "'eady" neighbor.
- Pop d from stack and mark it as "Visited". It has no "'eady" neighbor.
- Pop e from stack and mark it as "Visited". It has no "'eady" neighbor.
- Stack is empty. So stop.

So the traversal order is −

$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$

The alternate orders of traversal are −

$a \rightarrow e \rightarrow b \rightarrow c \rightarrow d$

Or, $a \rightarrow b \rightarrow e \rightarrow c \rightarrow d$

Or, $a \rightarrow d \rightarrow e \rightarrow b \rightarrow c$

Or, $a \rightarrow d \rightarrow c \rightarrow e \rightarrow b$

Or, $a \rightarrow d \rightarrow c \rightarrow b \rightarrow e$

Complexity Analysis - Let $G(V,E)$ be a graph with $|V|$ number of vertices and $|E|$ number of edges. If DFS algorithm visits every vertex in the graph and checks every edge, then the time complexity is −

$$\ominus (|V|+|E|)$$

Applications

1. Detecting cycle in a graph
2. To find topological sorting
3. To test if a graph is bipartite
4. Finding connected components
5. Finding the bridges of a graph
6. Finding biconnectivity in graphs
7. Solving the Knight's Tour problem
8. Solving puzzles with only one solution

Subject Notes
Discrete Structures

UNIT-5

Posets, Hasse Diagram and Lattices:

Introduction: Elementary mathematics tends to focus lopsidedly on computational structures. Relational ideas have become more important with the advent of computer science and the rise of discrete mathematics, however. Many contemporary mathematical applications involve binary or n-ary relations in addition to computations. In this chapter we will explore other kinds of relations (these will all be binary relations here), particularly ones that impose an order of one sort or another on a set. This will lead us to investigate certain order-structures (posets, lattices) and to introduce an abstract type of algebra known as Boolean algebra. Our exploration of these ideas will nicely tie together some earlier ideas in logic and set theory as well as lead us into areas that are of crucial importance to computer science.

Ordered set - If a set X is ordered in a reasonable way, then there is a natural way to define an "order topology" on X. Most interesting (for our purposes) will be ordered sets that satisfy a very strong ordering condition: that every nonempty subset contains a smallest element. Such sets are called well-ordered. The most familiar example of a well-ordered set is N.

Partially Ordered Set (POSET) - A partially ordered set consists of a set with a binary relation which is reflexive, antisymmetric and transitive. "Partially ordered set" is abbreviated as POSET.

Examples

1. **The set of real numbers under binary operation less than or equal to (≤) is a poset.**

Solution - Let the set *S*= {1, 2, 3} and the operation is ≤
The relations will be {(1,1),(2,2),(3,3),(1,2),(1,3),(2,3)}
This relation R is **reflexive** as {(1, 1), (2, 2), (3, 3)} ∈*R*
This relation R is **antisymmetric**, as {(1, 2),(1,3),(2,3)}∈*R and* {(1,2),(1,3),(2,3)}ø*R*
This relation R is also **transitive** as {(1, 2), (2, 3), (1, 3)} ∈ *R*
Hence, it is a **poset**.
Note: The vertex set of a directed acyclic graph under the operation 'reachability' is a poset.

Hasse Diagram of Partially Ordered Set - The Hasse diagram of a poset is the directed graph whose vertices are the element of that poset and the arcs covers the pairs (x, y) in the poset. If in the poset*x*<*y*, then the point x appears lower than the point y in the Hasse diagram. If *x*<*y*<*z*in the poset, then the arrow is not shown between x and z as it is implicit.

Example

The poset of subsets of {1,2,3}={Ø,{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}} is shown by the following Hasse diagram –

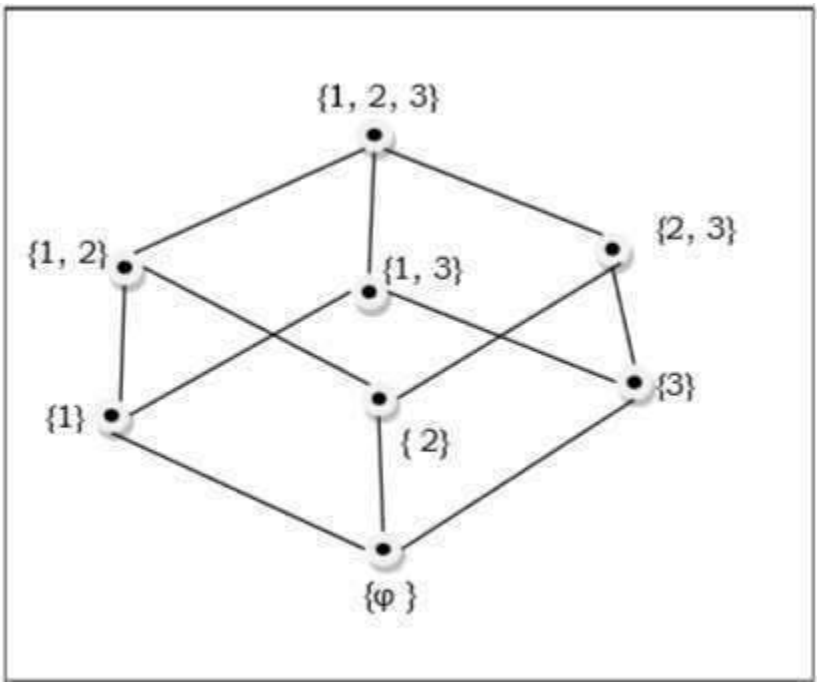


Figure 5.1 Hasse diagram

Isomorphic ordered set - Two partially ordered sets are said to be isomorphic if their "structures" are entirely analogous. Formally, partially ordered sets P=(X, <=) and Q=(X', <=') are isomorphic if there is a bijection f from X to X' such that x1<=x2 precisely when f(x1) <='f(x2).

Well-ordered set- A well-ordered set is a totally ordered set in which every nonempty subset has a least member.

An example of well-ordered set is the set of positive integers with the standard order relation ($\mathbb{Z}^{+)}$ because any nonempty subset of it has least member. However, $\mathbb{R}^{+}$ (the positive reals) is not a well-ordered set with the usual order, because (0, 1) = {x: 0 < x < 1} is a nonempty subset but it doesn't contain at least number.

A well-ordering of a set X is the result of defining a binary relation ≤ on X to itself in such a way that X becomes well-ordered with respect to ≤.

Linearly Ordered Set - A Linearly ordered set or Total ordered set is a partial order set in which every pair of element is comparable. The elements $a, b \in S$ are said to be comparable if either $a \leq b$ or $b \leq a$ holds. Trichotomy law defines this total ordered set. A totally ordered set can be defined as a distributive lattice having the property $\{a \vee b, a \wedge b\} = \{a, b\}$ for all values of a,b in set S.

Lattice - A lattice is a poset $(L, \leq)$ for which every pair $\{a, b\} \in L$ has a least upper bound (denoted by $a \vee b$) and a greatest lower bound (denoted by $a \wedge b$). LUB ($\{a, b\}$) is called the join of a and b. GLB ($\{a, b\}$) is called the meet of a and b.

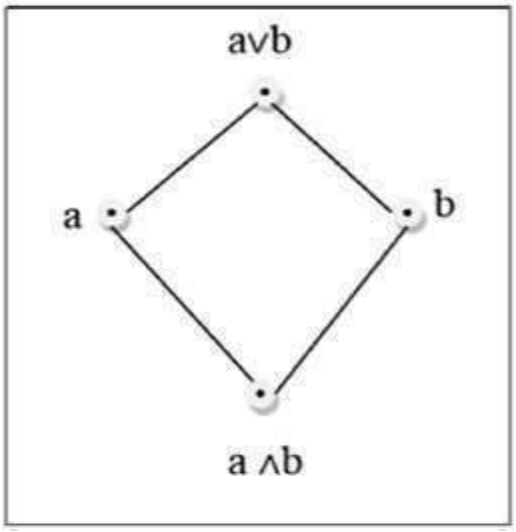


Figure 5.2 **Lattice**

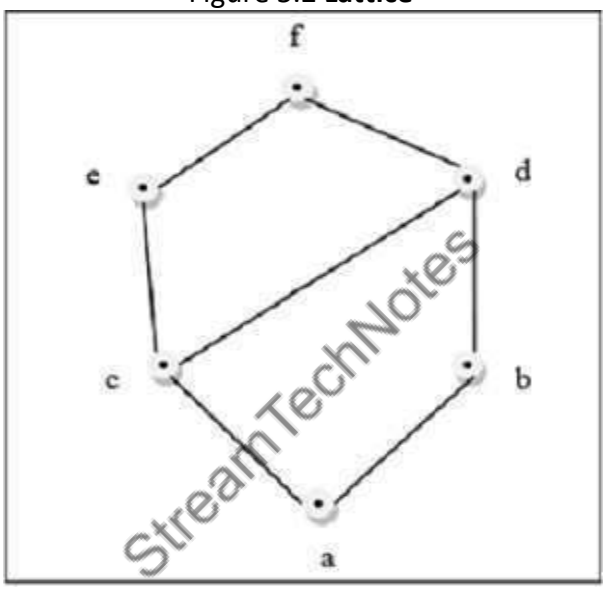


Figure 5.3 example of lattice

Figure 5.3 is a lattice because for every pair {*a*, *b*} ∈*L*, a GLB and a LUB exists.

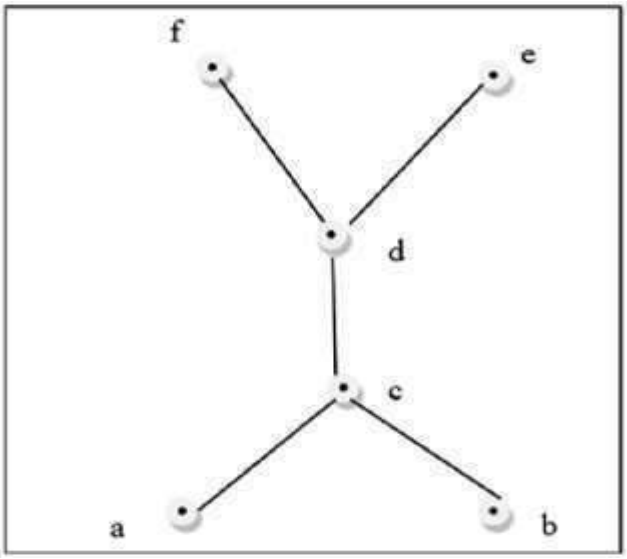


Figure 5.4 example of lattice

Figure 5.4 is a not a lattice because *GLB* (*a*, *b*) and *LUB* (*e*, *f*) does not exist.

Properties of Lattices

1. **Idempotent Properties**
   a) $a \vee a=a$
   b) $a \wedge a=a$
2. **Absorption Properties**
   a) $a \vee (a \wedge b)=a$
   b) $a \wedge (a \vee b)=a$
3. **Commutative Properties**
   a) $a \vee b=b \vee a$
   b) $a \wedge b=b \wedge a$
4. **Associative Properties**
   a) $a \vee (b \vee c)=(a \vee b) \vee c$
   b) $a \wedge (b \wedge c)=(a \wedge b) \wedge c$

Dual of a Lattice - The dual of a lattice is obtained by interchanging the 'v' and 'A' operations.

Example

The dual of $[a \vee (b \wedge c)]$ *is* $[a \wedge (b \vee c)]$

Types of Lattices

1. **Bounded Lattice -** A lattice L becomes a bounded lattice if it has a greatest element 1 and a least element 0.
2. **Complemented Lattice -** A lattice L becomes a complemented lattice if it is a bounded lattice and if every element in the lattice has a complement. An element x has a complement x' if $\exists x(x \wedge x'=0$ *and* $x \vee x'=1)$
3. **Distributive Lattice -** If a lattice satisfies the following two distribute properties, it is called a distributive lattice.
   a) $a \vee (b \wedge c)=(a \vee b) \wedge (a \vee c)$
   b) $a \wedge (b \vee c)=(a \wedge b) \vee (a \wedge c)$
4. **Modular Lattice - If** a lattice satisfies the following property, it is called modular lattice.
   $a \wedge (b \vee (a \wedge d)) = (a \wedge b) \vee (a \wedge d)$
5. **Isomorphic Lattices** If f: L1 -> L2 is an isomorphism from the poset (L1, ≤1) to the poset (L2, ≤2) then L1 is a lattice iff L2 is a lattice. If a and b are elements of L1 then f(a Λ b) = f(a) Λ f(b) and f(a v b) = f(a) v f(b) If two lattices are isomorphic as posets we say they are isomorphic lattices.

Combinatorics: Combinatorics is the branch of mathematics studying the enumeration, combination, and permutation of sets of elements and the mathematical relations that characterize their properties. Mathematicians sometimes use the term "combinatorics" to refer to a larger subset of discrete mathematics that includes graph theory.

Counting Theory - In daily lives, many a times one needs to find out the number of all possible outcomes for a series of events. For instance, in how many ways can a panel of judges comprising of 6 men and 4 women be chosen from among 50 men and 38 women? How many different 10 lettered PAN numbers can be generated such that the first five letters are capital alphabets, the next four are digits and the last is again a capital letter. For solving these problems, mathematical theory of counting are used. **Counting** mainly encompasses fundamental counting rule, the permutation rule, and the combination rule.

The Rules of Sum and Product

The **Rule of Sum** and **Rule of Product** are used to decompose difficult counting problems into simple problems.

1. **The Rule of Sum** − If a sequence of tasks $T1,T2,\ldots,Tm$ can be done in $w1,w2,\ldots wm$ ways respectively (the condition is that no tasks can be performed simultaneously), then the number of ways to do one

of these tasks is $w1+w2+\cdots+wm$. If we consider two tasks A and B which are disjoint (i.e. $A \cap B=\emptyset$), then mathematically $|A \cup B|=|A|+|B|$

2. **The Rule of Product** − If a sequence of tasks $T1, T2, \ldots, Tm$ can be done in $w1, w2, \ldots wm$ ways respectively and every task arrives after the occurrence of the previous task, then there are $w1 \times w2 \times \cdots \times wm$ ways to perform the tasks. Mathematically, if a task B arrives after a task A, then $|A \times B|=|A| \times |B|$

Example

Question − A boy lives at X and wants to go to School at Z. From his home X he has to first reach Y and then Y to Z. He may go X to Y by either 3 bus routes or 2 train routes. From there, he can either choose 4 bus routes or 5 train routes to reach Z. How many ways are there to go from X to Z?

Solution − From X to Y, he can go in 3+2=5 ways (‘ule of Sum). Thereafter, he can go Y to) in 4+5=9 ways (Rule of Sum). Hence from X to Z he can go in 5×9=45 ways (Rule of Product).

Permutations - A permutation is an arrangement of some elements in which order matters. In other words a Permutation is an ordered Combination of elements.

Examples

1. From a set S ={x, y, z} by taking two at a time, all permutations are − *xy*, *yx*, *xz*, *zx*, *yz*, *zy*
2. We have to form a permutation of three digit numbers from a set of numbers S={1,2,3}

Different three digit numbers will be formed when we arrange the digits. The permutation will be = 123, 132, 213, 231, 312, 321

Number of Permutations

The number of permutations of ‘n’ different things taken ‘r’ at a time is denoted by ^nP_r

$$^nPr = n!/(n-r)!$$ Where $n!$ =1.2.3.... $(n-1).n$

Proof − Let there be ‘n’ different elements.

There is n number of ways to fill up the first place. After filling the first place (n-1) number of elements is left. Hence, there are (n-1) ways to fill up the second place. After filling the first and second place, (n-2) number of elements is left. Hence, there are (n-2) ways to fill up the third place. We can now generalize the number of ways to fill up r-th place as [n – (r–1)] = n–r+1

So, the total no. of ways to fill up from first place up to r-th-place –

$^nPr = n(n-1)(n-2)\ldots(n-r+1)$

$= [n(n-1)(n-2)\ldots(n-r+1)][(n-r)(n-r-1)\ldots 3.2.1]/[(n-r)(n-r-1)\ldots 3.2.1]$

Hence,

$^nPr = n!/(n-r)!$

Some important formulas of permutation

1. If there are n elements of which $a1$ are alike of some kind, $a2$ are alike of another kind; $a3$ are alike of third kind and so on and ar are of rth kind, where $(a1+a2+\ldots ar)=n$ Then, number of permutations of these n objects is = $n!/[(a1!(a2!)\ldots (ar!)]$
2. Number of permutations of n distinct elements taking n elements at a time = $nPn = n!$
3. The number of permutations of n dissimilar elements taking r elements at a time, when x particular things always occupy definite places = $^{n-x}p_{r-x}$
4. The number of permutations of n dissimilar elements when r specified things always come together is − $r!\,(n-r+1)!$
5. The number of permutations of n dissimilar elements when r specified things never come together is − $n!-[r!\,(n-r+1)!]$
6. The number of circular permutations of n different elements taken x elements at time = $^np_x/x$
7. The number of circular permutations of n different things = $^np_n/n$

Some Problems

Problem 1 − from a bunch of 6 different cards, how many ways we can permute it?
Solution − as we are taking 6 cards at a time from a deck of 6 cards, the permutation will be ${}^{6}P6=6! =720$
Problem 2 − in how many ways can the letters of the word ''EADE'' be arranged?
Solution − There are 6 letters word (2 E, 1 A, 1D and 2'.) in the word ''EADE''.
The permutation will be =6! / [(2!)(1!)(1!)(2!)]=180.
Problem 3 − in how ways can the letters of the word 'O'ANGE' be arranged so that the consonants occupy only the even positions?
Solution − There are 3 vowels and 3 consonants in the word 'O'ANGE'. Number of ways of arranging the consonants among themselves $={}^{3}P3=3! =6$.
The remaining 3 vacant places will be filled up by 3 vowels in ${}^{3}P3=3! =6$ ways. Hence, the total number of permutation is 6×6=36

Combinations − A combination is selection of some given elements in which order does not matter.The number of all combinations of n things, taken r at a time is −

$$^{n}C\ r= n! /(r! (n - r)!)$$

Problem 1 - Find the number of subsets of the set {1, 2, 3, 4, 5, 6} having 3 elements.
Solution - The cardinality of the set is 6 and we have to choose 3 elements from the set. Here, the ordering does not matter. Hence, the number of subsets will be ${}^{6}C3=20$
Problem 2 - There are 6 men and 5 women in a room. In how many ways we can choose 3 men and 2 women from the room?
Solution - The number of ways to choose 3 men from 6 men is ${}^{6}C3$ and the number of ways to choose 2 women from 5 women is ${}^{5}C2$
Hence, the total number of ways is − ${}^{6}C3\times{}^{5}C2=20\times10=200$
Problem 3 - How many ways can you choose 3 distinct groups of 3 students from total 9 students?
Solution - Let us number the groups as 1, 2 and 3
For choosing 3 students for 1st group, the number of ways − ${}^{9}C3$
The number of ways for choosing 3 students for 2nd group after choosing 1st group − ${}^{6}C3$
The number of ways for choosing 3 students for 3rd group after choosing 1st and 2nd group − ${}^{3}C3$
Hence, the total number of ways $={}^{9}C3\times{}^{6}C3\times{}^{3}C3=84\times20\times1=1680$

Pascal's Identity
Pascal's identity, first derived by Blaise Pascal in 19th century, states that the number of ways to choose k elements from n elements is equal to the summation of number of ways to choose (k-1) elements from (n-1) elements and the number of ways to choose elements from n-1 elements.
Mathematically, for any positive integers k and n: ${}^{n}C\ k= {}^{n-1}C_{k-1}+{}^{n-1}C_{k}$
Proof −
${}^{n-1}C_{k-1}+{}^{n-1}C_{k}$
$= (n-1)!\ (k-1)!\ (n-k)! + (n-1)!k!(n-k-1)!$
$= (n-1)!\ (kk!(n-k)!+n-kk!(n-k)!)$
$= (n-1)!\ nk!(n-k)!$
$= n!k!(n-k)!$
$=nCk$

Probability - Closely related to the concepts of counting is Probability. We often try to guess the results of games of chance, like card games, slot machines, and lotteries; i.e. we try to find the likelihood or probability that a particular result with be obtained.
Probability can be conceptualized as finding the chance of occurrence of an event. Mathematically, it is the study of random processes and their outcomes. The laws of probability have a wide applicability in a variety of fields like genetics, weather forecasting, opinion polls, stock markets etc.

Basic Concepts

Probability theory was invented in the 17th century by two French mathematicians, Blaise Pascal and Pierre de Fermat, who were dealing with mathematical problems regarding of chance.

Before proceeding to details of probability, let us get the concept of some definitions.

Random Experiment − an experiment in which all possible outcomes are known and the exact output cannot be predicted in advance is called a random experiment. Tossing a fair coin is an example of random experiment.

Sample Space − When we perform an experiment, then the set S of all possible outcomes is called the sample space. If we toss a coin, the sample space $S = \{H, T\}$

Event − any subset of a sample space is called an event. After tossing a coin, getting Head on the top is an event.

The word "probability" means the chance of occurrence of a particular event. The best we can say is how likely they are to happen, using the idea of probability.

Probability of occurrence of an event=Total number of favorable outcome total number of outcomes

As the occurrence of any event varies between 0% and 100%, the probability varies between 0 and 1.

Steps to find the probability

Step 1 − Calculate all possible outcomes of the experiment.

Step 2 − Calculate the number of favorable outcomes of the experiment.

Step 3 − Apply the corresponding probability formula.

Tossing a Coin

If a coin is tossed, there are two possible outcomes − Heads (H) or Tails (T)

So, Total number of outcomes = 2

Hence, the probability of getting a Head (H)

on top is 1/2 and the probability of getting a Tails (T)

on top is 1/2

Throwing a Dice

When a dice is thrown, six possible outcomes can be on the top − 1, 2, 3, 4, 5, 6.

The probability of any one of the numbers is 1/6

The probability of getting even numbers is 3/6 = 1/3

The probability of getting odd numbers is 3/6 = 1/3

Taking Cards from a Deck

From a deck of 52 cards, if one card is picked find the probability of an ace being drawn and also find the probability of a diamond being drawn.

Total number of possible outcomes − 52

Outcomes of being an ace − 4

Probability of being an ace = 4/52 = 1/13

Probability of being a diamond = 4/52 = 1/13

Probability Axioms

1. The probability of an event always varies from 0 to 1. $[0 \le P(x) \le 1]$
2. For an impossible event the probability is 0 and for a certain event the probability is 1.
3. If the occurrence of one event is not influenced by another event, they are called mutually exclusive or disjoint.
4. If $A1, A2 \ldots An$ are mutually exclusive/disjoint events, then $P(Ai \cap Aj) = \emptyset$ for $I \neq j$ and $P(A1 \cup A2 \cup \ldots An) = P(A1) + P(A2) + \ldots P(An)$

Properties of Probability

1. If there are two events x and x^{---} which are complementary, then the probability of the complementary event is −

$p(x^{---}) = 1 - p(x)$

2. For two non-disjoint events A and B, the probability of the union of two events −

$P(A \cup B) = P(A) + P(B)$

3. If an event A is a subset of another event B (i.e. $A \subset B$), then the probability of A is less than or equal to the probability of B. Hence, $A \subset B$ implies $P(A) \leq p(B)$

Conditional Probability

The conditional probability of an event B is the probability that the event will occur given an event A has already occurred. This is written as $P(B|A)$.

Mathematically − $P(B|A) = P(A \cap B)/P(A)$

If event A and B are mutually exclusive, then the conditional probability of event B after the event A will be the probability of event B that is $P(B)$

Problem 1 - In a country 50% of all teenagers own a cycle and 30% of all teenagers own a bike and cycle. What is the probability that a teenager owns bike given that the teenager owns a cycle?

Solution - Let us assume A is the event of teenagers owning only a cycle and B is the event of teenagers owning only a bike.

So, $P(A) = 50/100 = 0.5$ and $P(A \cap B) = 30/100 = 0.3$ from the given problem.

$P(B|A) = P(A \cap B)/P(A) = 0.3/0.5 = 0.6$

Hence, the probability that a teenager owns bike given that the teenager owns a cycle is 60%.

Problem 2 - In a class, 50% of all students play cricket and 25% of all students play cricket and volleyball. What is the probability that a student plays volleyball given that the student plays cricket?

Solution - Let us assume A is the event of students playing only cricket and B is the event of students playing only volleyball.

So, $P(A) = 50/100 = 0.5$ and $P(A \cap B) = 25/100 = 0.25$ from the given problem.

$0.25/0.5 = 0.5$

Hence, the probability that a student plays volleyball given that the student plays cricket is 50%.

Problem 3 - Six good laptops and three defective laptops are mixed up. To find the defective laptops all of them are tested one-by-one at random. What is the probability to find both of the defective laptops in the first two pick?

Solution- Let A is the event that we find a defective laptop in the first test and B is the event that we find a defective laptop in the second test.

Hence, $P(A \cap B) = P(A)P(B|A) = 3/9 \times 2/8 = 1/21$

Bayes' Theorem

Theorem − If A and B are two mutually exclusive events, where $P(A)$ is the probability of A and $P(B)$ is the probability of B, $P(A|B)$ is the probability of A given that B is true. $P(B|A)$ is the probability of B given that A is true, then Bayes' Theorem states −

$$\boldsymbol{P(A|B) = P(B|A)P(A) \sum ni = 1 P(B|Ai)P(Ai)}$$

Application of Bayes' Theorem

1. in situations where all the events of sample space are mutually exclusive events.

2. In situations where either $P(Ai \cap B)$ for each Ai or $P(Ai)$ and $P(B|Ai)$ for each Ai is known.

Problem - Consider three pen-stands. The first pen-stand contains 2 red pens and 3 blue pens; the second one has 3 red pens and 2 blue pens; and the third one has 4 red pens and 1 blue pen. There is equal probability of each pen-stand to be selected. If one pen is drawn at random, what is the probability that it is a red pen?

Solution - Let Ai be the event that i^{th} pen-stand is selected. Here, i = 1, 2, 3.

Since probability for choosing a pen-stand is equal, $P(Ai) = 1/3$

Let B be the event that a red pen is drawn.

The probability that a red pen is chosen among the five pens of the first pen-stand,

$P(B|A1) = 2/5$

The probability that a red pen is chosen among the five pens of the second pen-stand,

$P(B|A2) = 3/5$
The probability that a red pen is chosen among the five pens of the third pen-stand,
$P(B|A3) = 4/5$
According to Bayes' Theorem,
$P(B)=P(A1).P(B|A1)+P(A2).P(B|A2)+P(A3).P(B|A3)$
$=1/3.2/5+1/3.3/5+1/3.4/5$
$=3/5$

Binomial Probability Distribution

Binomial Distribution It is a discrete probability distribution.
Binomial Probability is calculated by following general formula-
$P(X) = {}^{n}C_{x} p^{x} q^{(n-x)}$
Where,
n = number of trials
x = number of success
p = Probability of success
q = Probability of failure = 1 – p.

Requirements for a Binomial Distribution

- Random Experiment must involve n identical trials.
- As the word "Binomial" suggests, each trial should have only 2 possible outcomes, denoted as "Success" or "Failure".
- Each trial is independent of the previous trials.
- The probability of success denoted by p does not change from trial to trial.
- The probability of failure is 1-p and it is also fixed from trial to trial.

Sample question: "60% of people who purchase sports cars are men. If 10 sports car owners are randomly selected, find the probability that exactly 7 are men."

Step 1: Identify 'n' and 'X' from the problem.
Using our sample question, n (the number of randomly selected items — in this case, sports car owners are randomly selected) is 10, and X (the number you are asked to "find the probability" for) is 7.
Step 2: Figure out the first part of the formula, which is:
n! / (n – X)! X!
Substituting the variables:
10! / ((10 – 7)! × 7!)
This equals to 120
Set this number aside for a moment.
Step 3: Find "p" the probability of success and "q" the probability of failure.
We are given p = 60%, or .6.
Therefore, the probability of failure is 1 – .6 = .4 (40%).
Step 4: Work the next part of the formula. pX = .67 = .0.0279936 Set this number aside while you work the third part of the formula.
Step 5: Work the third part of the formula.
q(.4 – 7) =
.4(10-7) =
.43 = 0.064
Step 6: Multiply the three answers from steps 2, 4 and 5 together.
120 × 0.0279936 × 0.064
= 0.215.

Recurrence Relation - It Show how recursive techniques can derive sequences and be used for solving

counting problems. The procedure for finding the terms of a sequence in a recursive manner is called recurrence relation. We study the theory of linear recurrence relations and their solutions. Finally, we introduce generating functions for solving recurrence relations.

Definition

A recurrence relation is an equation that recursively defines a sequence where the next term is a function of the previous terms (Expressing *Fn* as some combination of *Fi* with $i<n$).

Example − Fibonacci series − $F_n = F_{n-1}+F_{n-2}$, Tower of Hanoi − $F_n=2F_{n-1}+1$

Recursive algorithms:

- All of the algorithms that have been developed so far have been iterative - they perform their task by repeating and following some statements in such a way that a task is achieved.
- Recursion is the process of defining something in terms of itself. Usually, the algorithm will attempt to simply the problem at hand by merely getting closer to the final result. The process is repeated until the problem has simplified enough so that it can be directly evaluated.

Example:

T (n) = T (n-1) + 1

T (0) = 1

Solving:

T (n-1) = T (n-2) + 1,

So T (n) = T (n-1) +1 = [T (n-2) + 1] + 1 = T (n-2) + 2,

And T (n-2) = T (n-3) +1

So T (n) = T (n-2) + 2 = [T (n-3) + 1] +1 = T (n-3) + 3,

And in general T (n) = T (n-i) + i;

So finally T (n) = T (0) + n = 1 + n

Linear Recurrence Relations with constant coefficients

A linear recurrence equation of degree k or order k is a recurrence equation which is in the format $xn= A_1x_n-1+A_2x_{n-1}+A_3x_{n-1}+...A_kx_{n-k}$ (A_n is a constant and $A_k\neq0$) on a sequence of numbers as a first-degree polynomial.

These are some examples of linear recurrence equations –

Recurrence relations	Initial values	Solutions
$F_n = F_{n-1} + F_{n-2}$	$a_1 = a_2 = 1$	Fibonacci number
$F_n = F_{n-1} + F_{n-2}$	$a_1 = 1, a_2 = 3$	Lucas Number
$F_n = F_{n-2} + F_{n-3}$	$a_1 = a_2 = a_3 = 1$	Padovan sequence
$F_n = 2F_{n-1} + F_{n-2}$	$a_1 = 0, a_2 = 1$	Pell number

Homogeneous Recurrence Relation: A recurrence relation is called homogeneous if it is in the form $F_n= AF_{n-1}+BF_{n-2}$

How to solve Homogeneous linear recurrence relation (Homogeneous solution)

Suppose, a two ordered linear recurrence relation is − $F_n = AF_{n-1}+BF_{n-2}$ where A and B are real numbers.

The characteristic equation for the above recurrence relation is −

$x^2-Ax-B=0$

Three cases may occur while finding the roots −

Case 1 − If this equation factors as $(x-x_1)$ $(x-x1) = 0$ and it produces two distinct real roots $x1$ and $x2$, then Fn =

$ax_1^n+bx_2^n$ is the solution. [Here, a and b are constants]

Case 2 − If this equation factors as $(x-x1)^2=0$ and it produces single real root x_1, then $F_n = ax_1^n+bnx_2^n$ is the solution.

Case 3 − If the equation produces two distinct complex roots, $x1$ and $x2$ in polar form $x1=r\angle\theta$ and $x2=r\angle(-\theta)$, then $F_n=r^n(a\cos(n\theta)+b\sin(n\theta))$ is the solution.

Problem 1 - Solve the recurrence relation $F_n = 5F_{n-1}-6F_{n-2}$ where $F0=1$ and $F1=4$

Solution - The characteristic equation of the recurrence relation is −

$x^2-5x+6=0$,

So, $(x-3)(x-2)=0$

Hence, the roots are −

$x1 = 3$ and $x2=2$

The roots are real and distinct. So, this is in the form of case 1

Hence, the solution is −

$Fn= ax^n{}_1+bx^n{}_2$

Here, $F_N = a3^n+b2^n$ (*As* $x1=3$ *and* $x2=2$)

Therefore,

$1=F_0=a3^0+b2^0= a+b$

$4=F_1=a3^1+b2^1=3a+2b$

Solving these two equations, we get $a=2$ and $b=-1$

Hence, the final solution is −

$\boldsymbol{F_n=2.3^n+(-1).^{2n}=2.3^n-2^n}$

Problem 2 Solve the recurrence relation − $F_n = 10\ F_{n-1}-25\ F_{n-2}$ where $F_0=3$ and $F_1=17$

Solution The characteristic equation of the recurrence relation is −

$x^2-10x + 25=0$

So $(x-5)^2=0$

Hence, there is single real root $x1=5$

As there is single real valued root, this is in the form of case 2

Hence, the solution is −

$F_N = ax_1^n+bnx_1^n$

$3=F0=a.5^0+b.0.5^0=a$

$17=F1=a.5^1+b.1.5^1=5a+5b$

Solving these two equations, we get $a=3$ and $b=2/5$

Hence, the final solution is − $F_n= 3.5^n+ (2/5).n.2^n$

Problem 3 Solve the recurrence relation $F_n = 2F_{n-1}-2F_{n-2}$ where $F_0=1$ and $F_1=3$

Solution The characteristic equation of the recurrence relation is −

$x^2-2x-2=0$

Hence, the roots are − $x1=1+i$and $x2=1-i$

In polar form,

$x1 = r\angle\vartheta$ and $x2=r\angle(-\vartheta)$, where $r=\sqrt{2}$ and $\vartheta=\pi4$

The roots are imaginary. So, this is in the form of case 3.

Hence, the solution is −

$Fn= (\sqrt{2})^n(a\cos(n.\sqcap/4)+b\sin(n.\sqcap/4))$

$1=F0= (\sqrt{2})^0(a\cos(0.\sqcap/4)+b\sin(0.\sqcap/4))=a$

$3=F1= (\sqrt{2})^1(a\cos(1.\sqcap/4)+b\sin(1.\sqcap/4))=\sqrt{2}(a/\sqrt{2}+b/\sqrt{2})$

Solving these two equations we get $a=1$ and $b=2$

Hence, the final solution is −

$Fn = (\sqrt{2})^n (cos(n.\pi/4)+2sin(n.\pi/4))$

Non-Homogeneous Recurrence Relation

A recurrence relation is called non-homogeneous if it is in the form

$F_n = AF_{n-1}+BF_{n-2}+f(n)$ where $f(n) \neq 0$

Particular solution: A recurrence relation **Fn**= $AF_{n-1}+BF_{n-2}+f(n)$ where $f(n) \neq 0$ Its associated **homogeneous recurrence** relation is $F_n = AF_{n-1}+BF_{n-2}$

The solution (a_n) of a non-homogeneous recurrence relation has two parts. First part is the solution (a_h) of the associated homogeneous recurrence relation and the second part is the **particular solution (a_t)**

$a_n = a_h + a_t$

To find the **particular solution**, we find an appropriate trial solution.

Let $f(n)=cx^n$; let $x^2=Ax+B$ be the characteristic equation of the associated homogeneous recurrence relation and let $x1$ and $x2$ be its roots.

a) If $x \neq x_1$ and $x \neq x_2$, then $a_t = Ax^n$

b) If $x = x_1$, $x \neq x_2$, then $a_t = Anx^n$

c) If $x = x_1 = x_2$, then $a_t = An^2x^n$

Total Solution: The total solution of linear recurrence relations with constant coefficients involves combining homogeneous and particular solution followed by determining constants.

Example

Let a non-homogeneous recurrence relation be $F_n = AF_{n-1}+BF_{n-2}+f(n)$ with characteristic roots $x_1=2$ and $x_2=5$. Trial solutions for different possible values of $f(n)$ are as follows −

f(n)	Trial solutions
4	A
5.2^n	$An2^n$
8.5^n	$An5^n$
4^n	$A4^n$
$2n^2+3n+1$	An^2+Bn+C

Problem:

Solve the recurrence relation $F_N = 3F_{n-1}+10F_{n-2}+7.5^n$ where $F_0=4$ and $F_1=3$

Solution:

This is a linear non-homogeneous relation, where the associated homogeneous equation is $F_N = 3F_{n-1}+10F_{n-2}$ and $f(n) = 7.5^n$

The characteristic equation of its associated homogeneous relation is −

$x^2-3x-10=0$

Or, $(x-5)(x+2) = 0$

Or, $x_1=5$ and $x_2=-2$

Hence $a_h = a.5^n+b.(-2)^n$, where a and b are constants. Since $f(n)=7.5^n$, i.e. of the form $c.x^n$, a reasonable trial solution of a_t will be Anx^n

$a_t = Anx^n = An5^n$

After putting the solution in the recurrence relation, we get −

$An5^n = 3A(n-1)5^{n-1}+10A(n-2)5^{n-2}+7.5^n$

Dividing both sides by 5^{n-2}, we get

$An5^2 = 3A(n-1)5+10A(n-2)5^0+7.5^2$

Or, $25An = 15An-15A+10An-20A+175$

Or, $35A = 175$

Or, $A=5$

So, $Fn = An5^n = 5n5^n = n5^{n+1}$

The solution of the recurrence relation can be written as −

$Fn = a_h + a_t = a.5^n + b.(-2)^n + n5^{n+1}$

Putting values of $F0=4$

and $F1=3$, in the above equation, we get $a=-2$ and $b=6$

Hence, the solution is −

$Fn = n5^{n+1} + 6.(-2)^n - 2.5n$

Generating Functions - Generating Functions represents sequences where each term of a sequence is expressed as a coefficient of a variable x in a formal power series.

Mathematically, for an infinite sequence, say $a0, a1, a2, \ldots, ak, \ldots$, the generating function will be −

$Gx = a0 + a1x + a2x^2 + \cdots + akx^k + \cdots = \sum akx^k (k=0 \text{ to } \infty)$

Some Areas of Application

Generating functions can be used for the following purposes −

a) For solving a variety of counting problems. For example, the number of ways to make change for a Rs. 100 note with the notes of denominations Rs.1, Rs.2, Rs.5, Rs.10, Rs.20 and Rs.50
b) For solving recurrence relations
c) For proving some of the combinatorial identities
d) For finding asymptotic formulae for terms of sequences

Solution by method of generating functions

Problem 1 - What are the generating functions for the sequences $\{a_k\}$ with $a_k=2$ and $a_k=3k$?

Solution When $a_k=2$, generating function, $G(x) = \sum(k=0 \text{ to } \infty)2x^k = 2+2x+2x^2+2x^3+\ldots$

When $a_k=3k$, $G(x) = \sum(k=0 \text{ to } \infty)3kx^k = 0+3x+6x^2+9x^3+\ldots\ldots$

Problem 2 - What is the generating function of the infinite series; 1, 1, 1, 1…?

Solution Here, $a_k=1$, for $0 \le k \le \infty$

Hence, $G(x) = 1+x+x^2+x^3+\ldots\cdots = 1/(1-x)$

Some Useful Generating Functions

a) For $a_k = a^k, G(x) = \sum(k=0 \text{ to } \infty)a^k x^k = 1+ax+a2x^2+\ldots\ldots\cdots = 1/(1-ax)$
b) For $a_k=(k+1), G(x) = \sum(k=0 \text{ to } \infty)(k+1)x^k = 1+2x+3x^2\ldots\ldots\cdots = 1(1-x)^2$
c) For $a_k = c^n{}_k, G(x) = \sum(k=0 \text{ to } \infty)c^n{}_k x^k = 1+c^n{}_1 x+c^n{}_2 x^2+\ldots\ldots\cdots+x^2 = (1+x)^n$
d) For $a_k=1/k!, G(x) = \sum(k=0 \text{ to } \infty)x^k/k! = 1+x+x^2/2!+x^2/3!\ldots\ldots\cdots = e^x$