Unit – III

3. Graph Theory

$G = (V, E)$; V = Vertices, E = Edges.

Graph is a collection of vertices and Edges.

Simple Graph

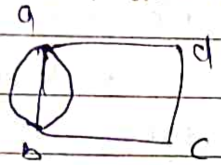
→ multiple edges or self loop not allowed.

~~Directed g~~

Multi Graph

→ multiple edges allowed.

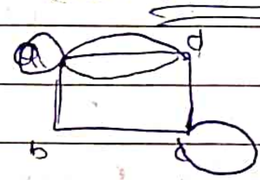
→ self loop not allowed.



Pseudo Graph

→ self loop allowed.

→ multiple edges allowed

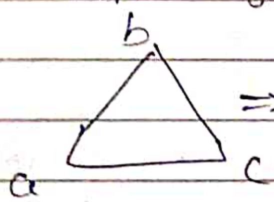


Directed graph — Undirected graph

↳ mixed graph. (having both ~~direct~~ directed and undirected)

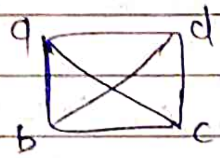
Complete graph

eg ①



⇒ K_3 graph.

②



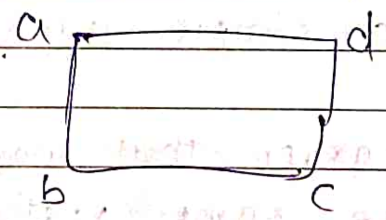
⇒ K_4 graph.

→ K_n = Complete graph.

n = no. of vertices.

Incomplete Graph

①

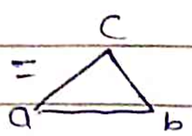


Cycle Graph.

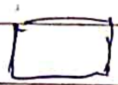
↳ C_n . $n \geq 3$

n = no. of vertices.

C_3 =



C_4 =



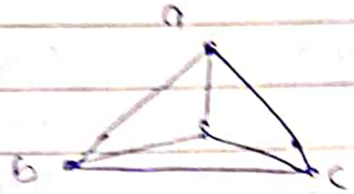
C_6 =

Vertex = Node.

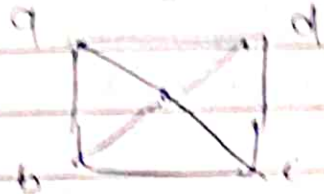
Wheel.

W_n , $n \geq 5$

W_3 =



W_4 =



Graph.

→ A Graph $G=(V, E)$ consists of
V = a non empty set of vertices or nodes, and
E = a set of edges.

→ Each edges has either one or two vertices associated with it called its end points.

→ A graph in which each edge connect two different vertices and where no two edges connect the same pair of vertices is called a simple graph.

→ A graph that may have multiple edges connecting the same vertices are called Multi graph.

→ Graph that may include loops and possibly multiple edges connecting the same pair of vertices are called Pseudo graph.

→ A graph with both directed and undirected edges is called mixed graph.

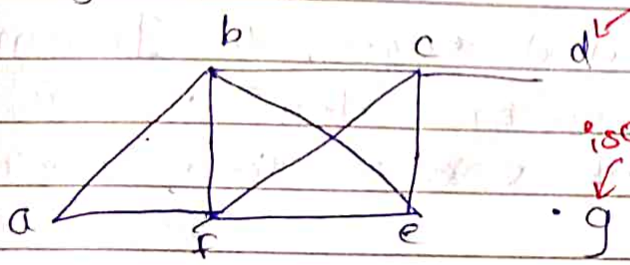
Degree of Vertex.

→ The degree of a vertex in an undirected graph is the number of edges incident with it except that a loop at a vertex contributes twice to the degree of that vertex (self loop: Add degree 2 not 1).

→ Degree of vertex 'V' is denoted by ⇒ deg(v)

eg. What are the degrees of the vertices in the graph given below.

(1)



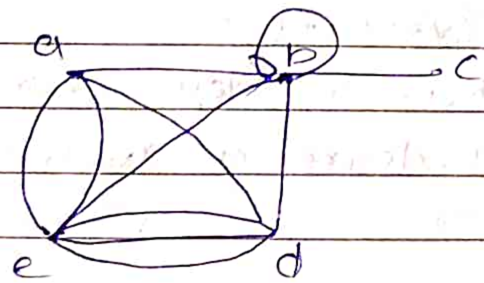
pendant

isolated

deg(a) = 2
deg(b) = 4
deg(c) = 4
deg(d) = 1
deg(e) = 3
deg(f) = 4
deg(g) = 0

Vertex with degree 0 = Isolated vertex.
Vertex with degree 1 = Pendant vertex

(2)



deg(a) = 4
deg(b) = ~~4~~ 6
deg(c) = 1
deg(d) = 4
deg(e) = 6

Let G = (V, E) be an undirected graph, with 'e' no. of edges then,

$$\boxed{2 \cdot e = \sum_{\in V} \deg(\ )}$$

↳ summation of degree of all the vertices

Un directed graph ⇒ for degree, self loop = +2

Directed graph ⇒ for degree, self loop = +1

Q1. How many edges are there in a graph with 10 vertices each of degree 6.

⇒

$$2 \times e = \sum_{\in V} \cancel{(10 \times 6)} \, (6+6+6+6+6+6+6+6+6+6)$$

↳ $6 \times 10 = 60$

$e = 30.$

Degree of Directed graph

→ In a graph with directed edges, the In degree of a vertex 'v' denoted by $\deg^-(v)$ is the no. of edges with v as their terminal vertex.

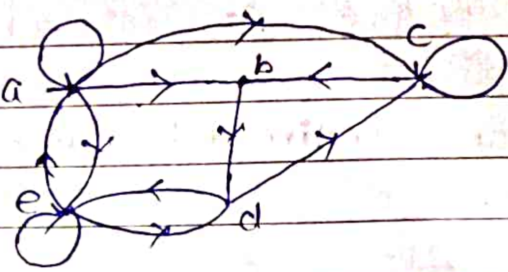
→ The Out degree of v denoted by $\deg^+(v)$ is the no. of edges with v as their initial vertex.

→ A loop at a vertex contribute 1 to both In degree and Out-degree of thes vertices.

→ A self loop is considered in both, i.e. in In degree as well as in Out degree.

eg. Find In-degree and out-degree of vertex of the given graph.

(1)



In-degree = No. of incomming edges towards that edge.

Out-degree = No. of Out going edges.

self loop = +1 (In

$\boxed{\text{Self loop = In degree as well as Out degree (+1)}}$

⇒ $\deg^-(a) = 2$.

in-degree of vertex a.

$\deg^-(b) = 2$

$\deg^-(c) = 3$

$\deg^-(d) = 2$

$\deg^-(e) = 3$

$\deg^+(a) = 4$

$\deg^+(b) = 1$

$\deg^+(c) = 2$

$\deg^+(d) = 2$

$\deg^+(e) = 3$

Complete Graph.

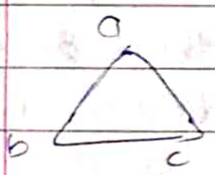
→ Complete graph on 'n' vertices denoted by 'K_n' is the simple graph that contains exact one edge betⁿ each pair of distinct vertices.

Bipartite Graph.

* ~~does~~ must not contain same elements in both sets.

→ A simple graph 'G' is called bipartite if its vertex set 'V' can be partion into two disjoint sets V_1 and V_2 such that, every edge in the graph connect ~~the~~ a vertex in V_1 and the vertex in V_2. (so that, no edge in 'G' connects either two vertices in V_1 or two vertices in V_2).

→ When this condⁿ holds, we call the pair (V_1, V_2) a Bipartition of the vertex set V of G.



$V = \{a, b, c\}$

$V_1 = \{a\}$ or $V_1 = \{a, b\}$

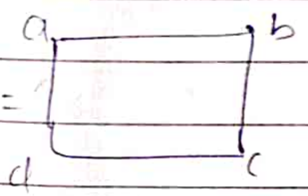
$V_2 = \{b, c\}$ $\quad$ $V_2 = \{c\}$

⇒ in graph b → c ~~graph~~ must not present

or a → b graph ~~must~~ not present

∴ this graph is not a Bipartite.

eg 1) C_4 =



$V = \{a, b, c, d\}$

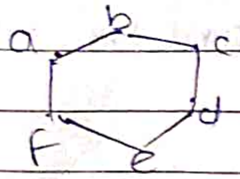
$V_1 = \{a, c\}$

$V_2 = \{b, d\}$

a → c
b → d } ⇒ graph not present

∴ It is a Bipartite graph

2) C_6 =



$V = \{a, b, c, d, e, f\}$

$V_1 = (a, c, e)$

$V_2 = \{f, b, d\}$

Complete Graph.

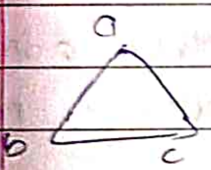
→ Complete graph on 'n' vertices denoted by 'K_n' is the simple graph that contains exact one edge betn each pair of distinct vertices

Bipartite Graph.

→ A simple graph 'G' is called bipartite if its vertex set 'V' can be partition into two disjoint sets V_1 and V_2 such that, every edge in the graph connect ~~the~~ a vertex in V_1 and the vertex in V_2. (so that, no edge in 'G' connects either two vertices in V_1 or two vertices in V_2).

* must not contein same elements in both sets.

→ When this condn holds, we call the pair (V_1, V_2) a Bipartition of the vertex set V of G.



$V = \{a, b, c\}$

$V_1 = \{a\}$ or $V_1 = \{a, b\}$

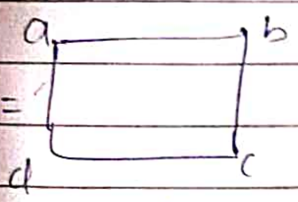
$V_2 = \{b, c\}$ $\quad V_2 = \{c\}$

⇒ in graph $b \to c$ graph must not present

or $a \to b$ graph ~~must~~ not present.

∴ this graph is not a Bipartite.

eg 1) C_4 =



$V = \{a, b, c, d\}$

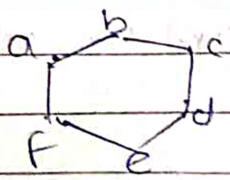
$V_1 = \{a, c\}$

$V_2 = \{b, d\}$

$a \to c$, $b \to d$ } ⇒ graph not present

∴ It is a Bipartite graph

2) C_6 =



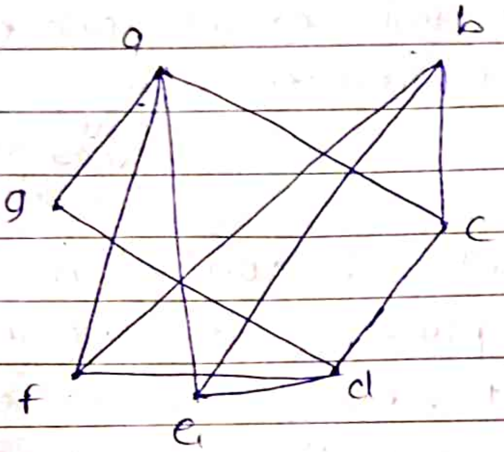
$V = \{a, b, c, d, e, f\}$

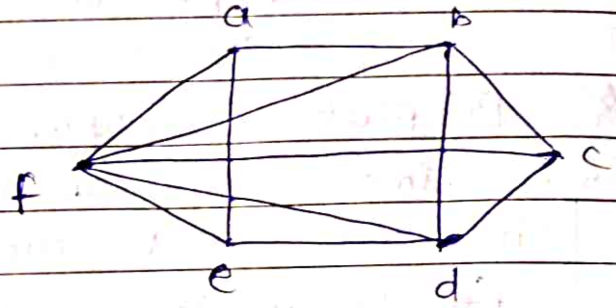
$V_1 = (a, c, e)$

$V_2 = \{f, b, d\}$

Q. Are the graph G and H displayed in the following fig. Bipartite.

→ G :- H :-





⇒

$V_G = \{a, b, c, d, e, f, g\}$

$V_1 = \{a, b, d\}$

$V_2 = \{g, f, e, c\}$

∵ V_1 and V_2 are disjoint

∴ Graph G is Bipartite

$V_H = \{a, b, c, d, e, f\}$

Graph H is not Bi-partite because, ~~$V_1 =$~~ its vertex set cannot be partion into two subsets, so that edges do not connect two vertices from the same subset.

Complete Bipartite graph.

→ It is denoted by $K_{m,n}$

set 1 contains 'm' no. of vertices

set 2 contains 'n' no. of vertices.

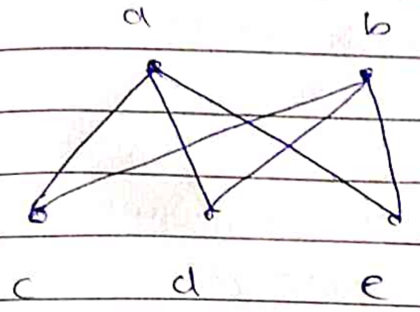
→ $K_{m,n}$ is the graph that has its vertex set partioned into two subset of 'm' and 'n' vertices respectively.

→ There is an edge betn two. vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

eg. 1) $K_{2,3}$

$V_1 = \{a, b\}$

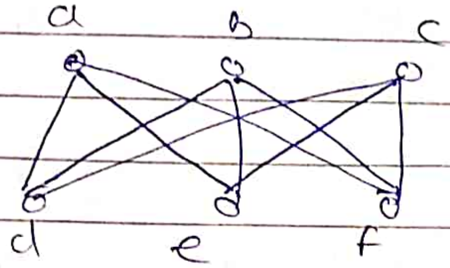
$V_2 = \{c, d, e\}$



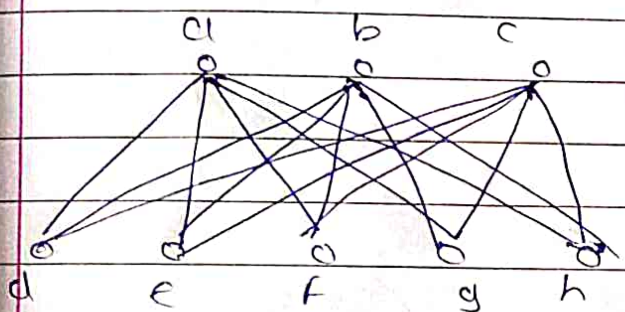
2) $K_{3,3}$

$V_1 = \{a, b, c\}$

$V_2 = \{d, e, f\}$



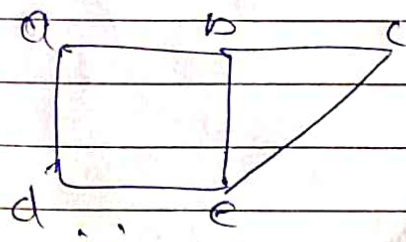
3) $K_{3,5}$ $V_1 = \{a, b, c\}$ $V_2 = \{d, e, f, g, h\}$



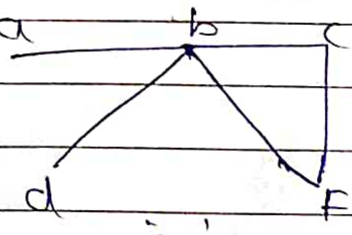
Union of Graph.

→ The union of two simple graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex ~~set~~ set $V_1 \cup V_2$ and $E_1 \cup E_2$.

→ union of G_1 & G_2 is denoted by '$G_1 \cup G_2$'.

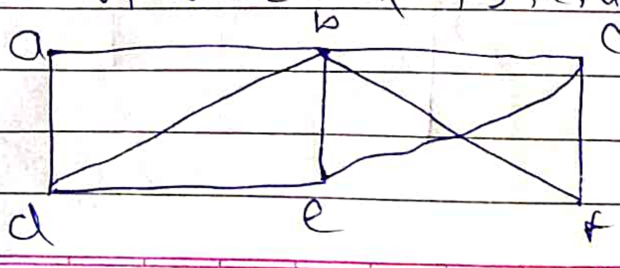


'G_1'



'G_2'

$V_1 \cup V_2 = \{a, b, c, d, e, f\}$



$= (G_1 \cup G_2)$.

Representing Graph / Graph Representation.

→ 3 ways.

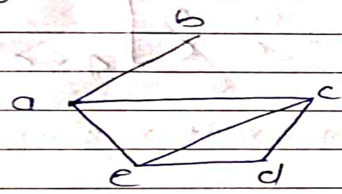
1) Adjacency list
2) Adjacency Matrices.
3) Incidence Matrices.

1) Adjacency list.

→ It specify the vertices that are adjacent to each vertex of the graph.

1) eg. Use adjacency list to describe the simple graph given below.

(1)



Adjacent list for the vertices.

vertices	Adjacent vertices.
a	b, c, e.
b	a
c	a, d, e
d	c, e
e	a, c, d

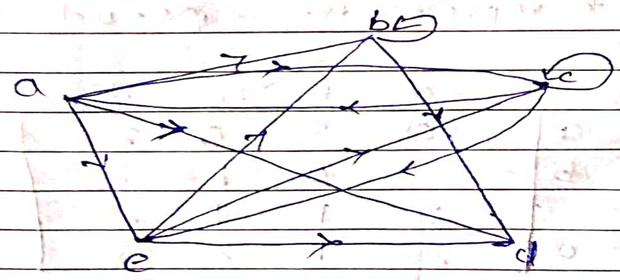
#

b → c

C = terminal vertex.

2) Represent the directed graph shown in the following fig. by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.

→



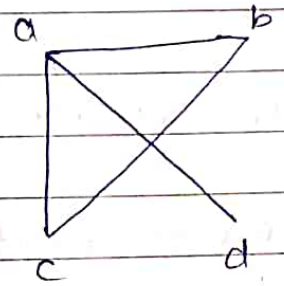
vertex	terminal vertex.
a	b, c, d, e.
b	d, b.
c	a, e, c.
d	—
e	b, c, d.

2) Adjacency Matrix.

→ The Adjacency matrix A of G (i.e. A_G) with respect to listing of the vertices is $n \times n$ 0,1 (zero-one) matrix with one as its $(i,j)^{th}$ entry when i and j are adjacent and zero as its $(i,j)^{th}$ entry when they are not adjacent.

① eg Use adjacency matrix to represent the graph shown below.

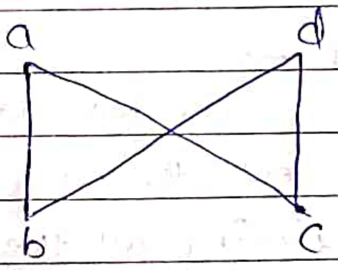
$|V| = 4 \Rightarrow \therefore 4 \times 4$ matrix.



$$\Rightarrow \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}_{4 \times 4} \end{array}$$

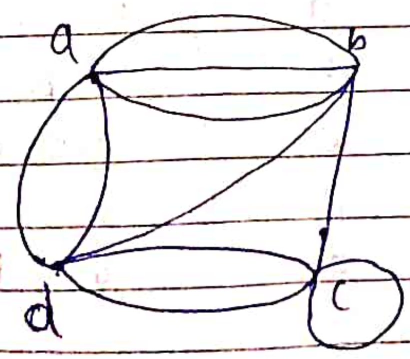
② Draw a graph with a adjacency matrix given below

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$



③ Use adjency matrix to represent the pseudo graph given below.

⇒



$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix} \end{array}$$

3) Incidence Matrix.

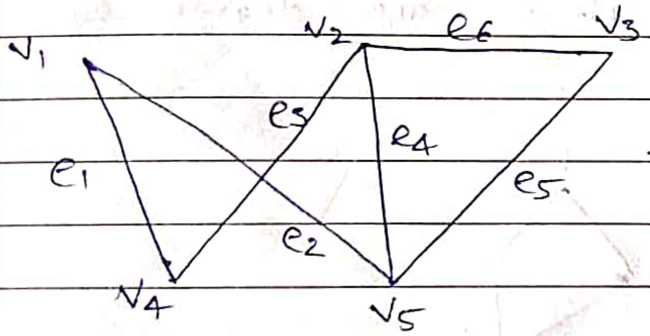
→ Let, $G = (V, E)$ be an undirected graph. suppose that $1, 2 \ldots n$ vertices and $E_1, E_2 \ldots E_m$ are edges of G ~~accorthen~~ then the incidence matrix w.r.t. this ordering of V and E is the '$n \times m$' matrix $M = [m_{i,j}]$

where, $m_{i,j} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } i \\ 0 & \text{otherwise} \end{cases}$

n = no. of vertices $\qquad m$ = no. of edges.

→ ~~eg~~. no. of rows = no. of vertices.
no. of columns = no. of edges. $\Big\}$ ~~m~~ $n \times m$.

eg. Represent the graph shown in following fig. with an incidence matrix.



$e_1 \Rightarrow v_1$ & $v_4 . = 1$

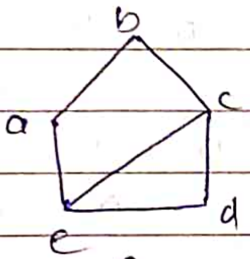
$$= \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccc} \boxed{e_1} & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}_{5 \times 6} \end{array}$$

Graph isomorphism.

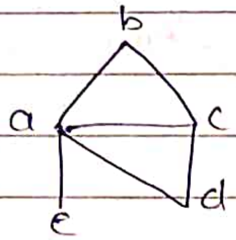
→ The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is one to one and onto function's 'f' from V_1 to V_2 with the property that, 'a' and 'b' are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2 ∀ a and b in V_1. such a fun 'f' is called isomorphism.

eg i) Determine whether the given graphs are isomorphic or not.

→ both graph:
1. must have same no. of vertices and 2. edges.
3. must have same degree and degree sequence
4. Mapping must be same.



G_1



G_2.

⇒ ① Here, vertices of G_1 = a, b, c, d, e = 5.
also vertices of G_2 = a, b, c, d, e = 5.
∴ Both graphs have same no. of vertices
∴ 1^{st} cond is true.

② Here, no. of edges of G_1 = 6
no. of edges of G_2 = 6
∴ Both graph have same no. of edges.
∴ 2^{nd} cond is true.

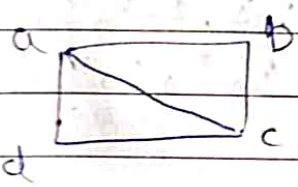
(3) Degree sequence.
G_1

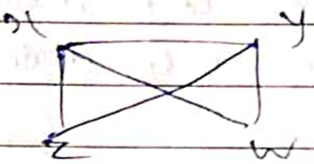
In G_1 degree of a, b, c, d, e = ~~...~~
in descending order => 3 3 2 2 2

In G_2.
degree of vertices in descending order = 4 3 2 2 1

∴ Here, degree sequence of both graphs are not same
∴ Given graphs are not isomorphic.

2) G_1 G_2.





=> (1) no. of vertices of G_1 = no. of vertices of G_2 = 4.
(2) no. of edges of G_1 = no. of edges of G_2 = 5

(3) Degree sequence of G_1 in descending order = 3, 3, 2, 2
Degree sequence of G_2 in descending order = 3, 3, 2, 2
Here 1st 3 cond's are true, check for 4th cond :-

(4) Mapping :-

a can be only mapped with same degree as a. (~~...~~ here a → x;
a → y;

deg(a) = 3	a →	x	deg(x) = 3
deg(b) = 2	b	z	deg(z) = 2
deg(c) = 3	c	~~...~~ y	deg(w) = 3
deg(d) = 2	d	~~...~~ w	deg(y) = ~~3~~ 2

let here $a \xrightarrow{\text{is mapped}} x$.

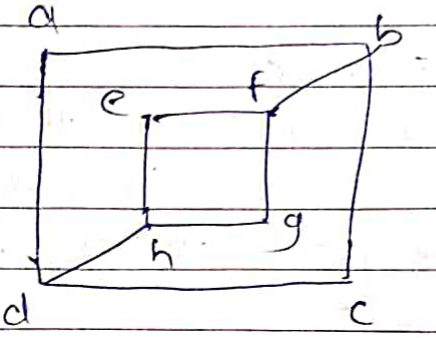
Similarly for all mapping.

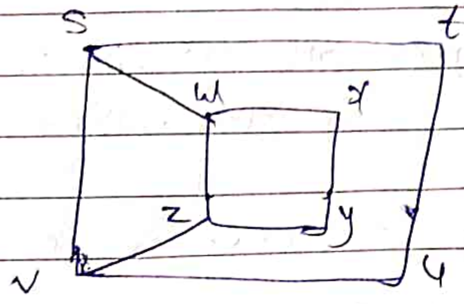
a → b $\overset{\text{must}}{\Rightarrow}$ x → z
a → c $\overset{\text{must}}{\Rightarrow}$ x → w.
a → d = yes $\overset{\text{must}}{\Rightarrow}$ x → y => yes.

similarly check for all cond's.

∴ It satisfies mapping cond => ∴ Graph is Isomorphic.

3) G_1 G_2





$\Rightarrow$ (1) no. of vertices are same.
(2) no. of edges are same.

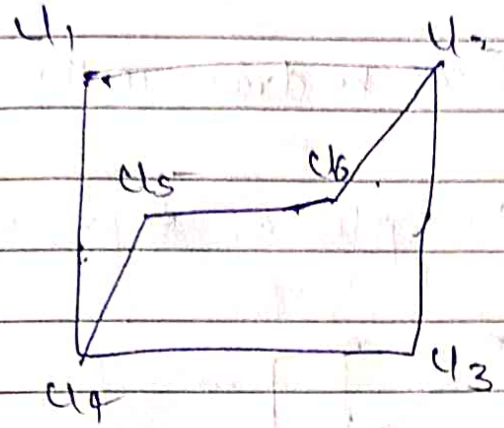
(3) degree sequence of G_1 in descending order : 3, 3, 3, 3, 2, 2, 2, 2
degree sequence of G_2 in descending order : 3, 3, 3, 3, 2, 2, 2, 2.

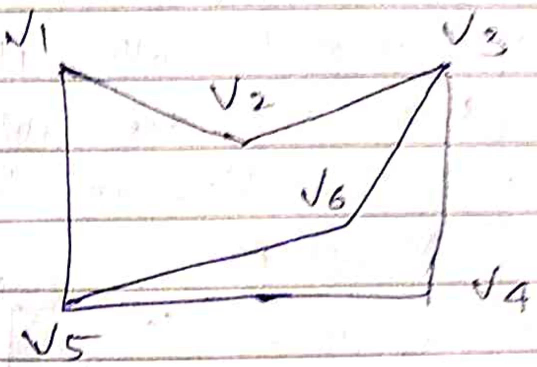
(4) Mapping :-

deg(a) = 2	a	t	deg(t) = 2
deg(b) = 3	b	s	deg(s) = 3
deg(c) = 2	c	u	deg(u) = 2
deg(d) = 3	d	v	deg(v) = 3
deg(e) = 2	e	x	deg(x) = 2
deg(f) = 3	f	w	deg(w) = 3
deg(g) = 2	g	y	deg(y) = 2
deg(h) = 3	h	z	deg(z) = 3

$\Rightarrow$ ~~It satisfies~~
It doesnot satisfies mapping cond.
$\therefore$ Graphs are not Isomorphus.

3) G_1 G_2





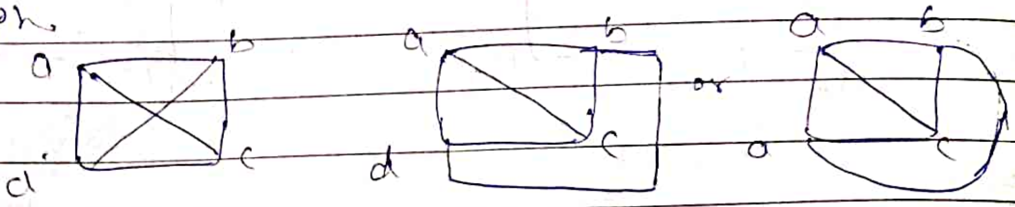
$\Rightarrow$

Planar Graph.

→ A graph is called planar if it can be drawn in the plane without edge crossing

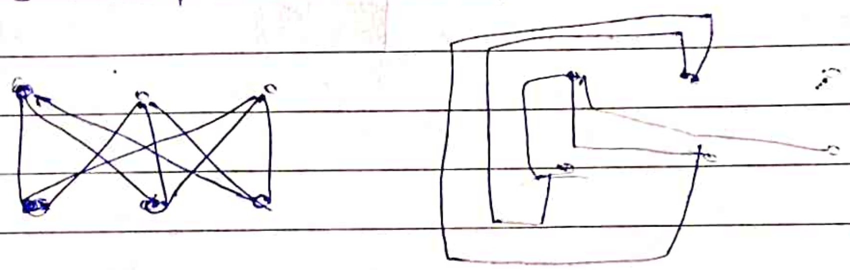
→ Such a drawing is called the planar representation of graph.

→ eg: K_4 =



non planar => planar.

→ K_3 complete bipartite graph.



=> This graph cannot be drawn such that it will be planar.

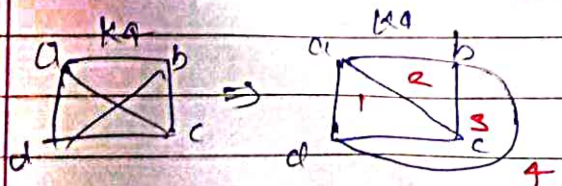
Euler Formula

→ Let 'G' be a connected planar simple graph with 'e' edges and 'v' vertices.

→ Let 'r' be the no. of regions in a planar representation of 'G', then,

→ eq $\boxed{r = e - v + 2}$

eg.



$\Rightarrow r = e - v + 2$

$= 6 - 4 + 2$

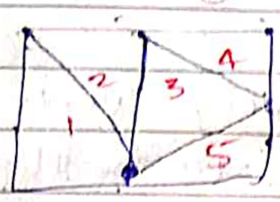
$r = 4$

Here '4' = outer region

Outer region = '1'.

Q. Find the no. of regions in the given graph.

⇒ 6



⇒ It is a planar graph.

$r = e - v + 2$

$= 11 - 7 + 2$

$= 4 + 2$

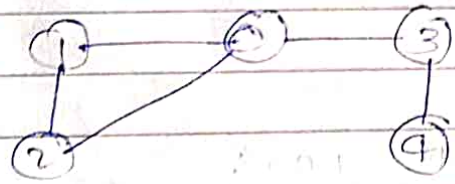
$r = 6$

★ Euler path and circuit.

→ Euler path is a path in graph that visit every edge exact once (No repetition of edges are allowed in euler path).

→ Euler circuit is an euler path that starts and ends on same vertex.

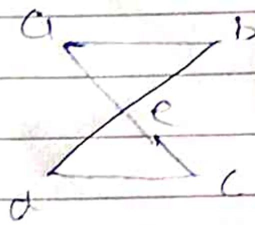
eg 1)



Euler path : 4 - 3 - 0 - 2 - 1 - 0 .

Here there is no euler circuit ∵ it cannot start and end on same vertex for euler path.

2)



⇒ euler path : a - b - e - c - d - e - a .

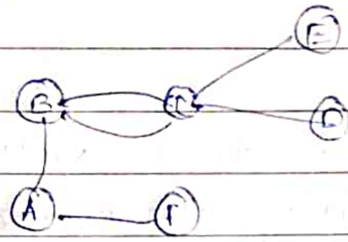
It is also a euler circuit ∵ first and last vertex is same.

Hamilton path and circuit.

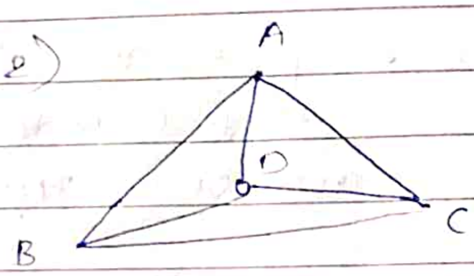
→ Hamilton path is a path that uses every vertx of the graph exat once. (No repeition of vertex is allowed).

→ A hamilton circuit is a hamilton path that begins and ends at the same vertex.

eg. 1)



2)

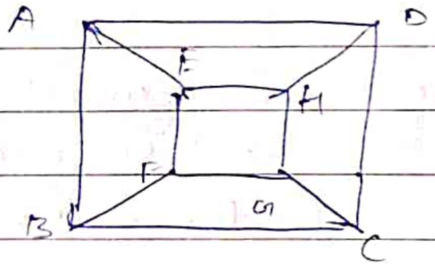


⇒ Not a Hamilton path

Hamilton path : ~~A - B - D - C - A~~.
A - B - C - D - A.

It is Hamilton circuit.

3)



⇒ Hamilton path :
A - B - F - G - C - D - H - E - A.
It is a Hamilton circuit.

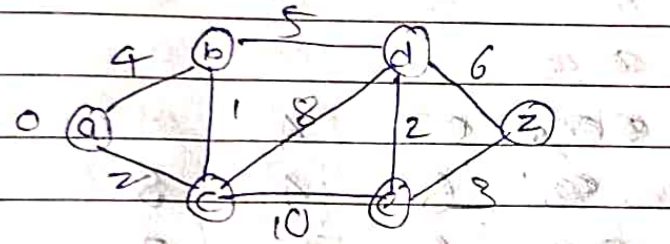
Imp.

Shortest Path Algorithm.

Single sourced shortest path.

Dijkstra's Algorithm.

eg. Find shortest path from A to z in the weighted graph shown below

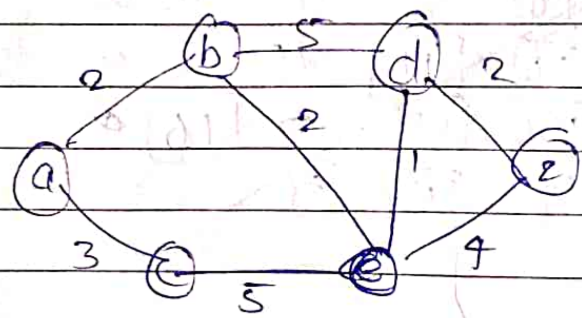


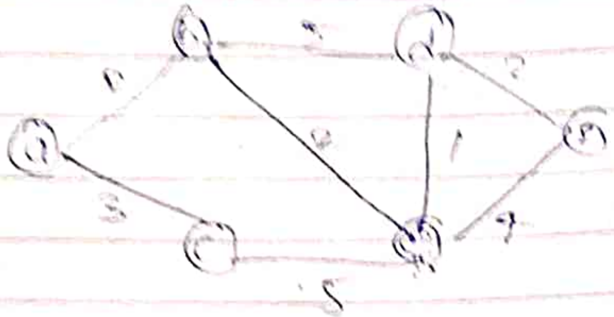
⇒

	b	c	d	e	z
a	∞	∞	∞	∞	∞
a	4	$\boxed{2}$	∞	∞	∞
a, c	$\boxed{3}$	$\boxed{2}$	10	12	∞
a, c, b	$\boxed{3}$	$\boxed{2}$	$\boxed{8}$	12	∞
a, c, b, d	$\boxed{3}$	$\boxed{2}$	$\boxed{8}$	$\boxed{10}$	14
a, c, b, d, e	$\boxed{3}$	$\boxed{2}$	$\boxed{8}$	$\boxed{10}$	$\boxed{13}$
a, c, b, d, e, z					

→ Hence, the shortest path is a, c, b, d, e, z (∵ its is = 13).

2) Find the length of shortest path betn a and z in the given weighted graph.

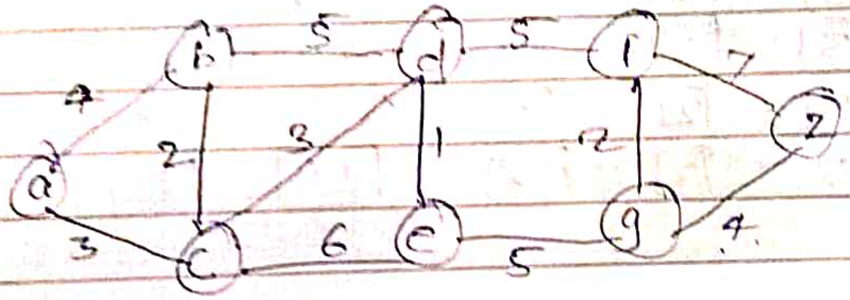




	b	c	d	e	z
a	∞	∞	∞	∞	∞
a	$\boxed{2}$	3	∞	∞	∞
a, b	$\boxed{2}$	[illegible]	7	$\boxed{4}$	∞
a, b, e	$\boxed{2}$	[illegible]	$\boxed{5}$	$\boxed{4}$	8
a, b, e, d	$\boxed{2}$	9	$\boxed{5}$	$\boxed{4}$	$\boxed{7}$
a, b, e, d, z					

shortest path : a, b, e, d, z
its length = 7

3)



→

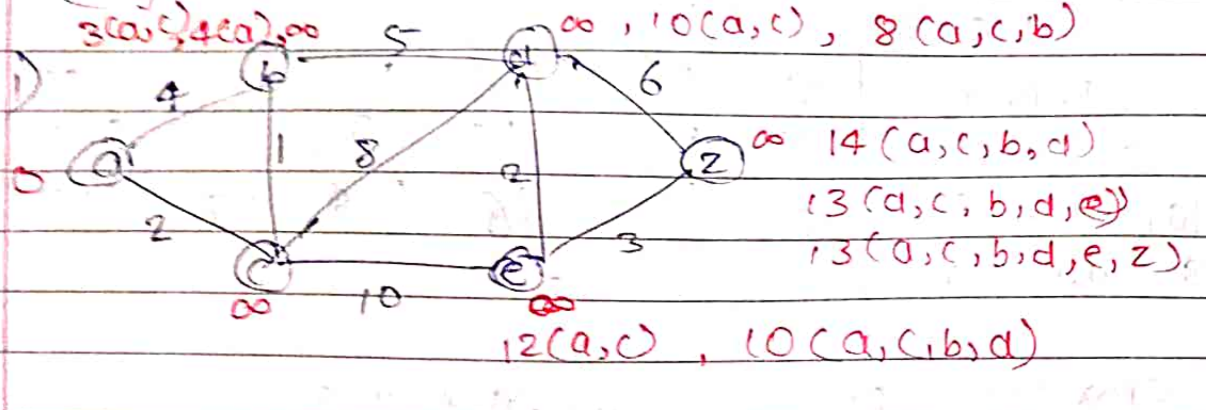
	b	c	d	e	f	g	z
a	∞	∞	∞	∞	∞	∞	∞
a	4	$\boxed{3}$	∞	∞	∞	∞	∞
a, c	$\boxed{5}$	$\boxed{3}$	6	9	∞	∞	∞
a, c, b	$\boxed{5}$	$\boxed{3}$	$\boxed{10}$	∞	∞	∞	∞
a, c, b, d	$\boxed{5}$	$\boxed{3}$	$\boxed{10}$	$\boxed{11}$	15	∞	∞
a, c, b, d, e	$\boxed{5}$	$\boxed{3}$	$\boxed{10}$	$\boxed{11}$	∞	$\boxed{16}$	∞
a, c, b, d, e	$\boxed{5}$	$\boxed{3}$	$\boxed{10}$	$\boxed{11}$	$\boxed{\infty}$		

	b	c	d	e	f	g	z
a	∞	∞	∞	∞	∞	∞	∞
a	4	$\boxed{3}$	∞	∞	∞	∞	∞
a, c	5	$\boxed{3}$	$\boxed{6}$	9	∞	∞	∞
a, c, d	5	$\boxed{3}$	$\boxed{6}$	$\boxed{7}$	11	∞	∞
a, c, d, e	5	$\boxed{3}$	$\boxed{6}$	$\boxed{7}$	∞	$\boxed{12}$	8
a, c, d, e, g	5	$\boxed{3}$	$\boxed{6}$	$\boxed{7}$	14	$\boxed{12}$	$\boxed{16}$
a, c, d, e, g, z							

Shortest path is: a, c, d, e, g, z.
its length = 16.

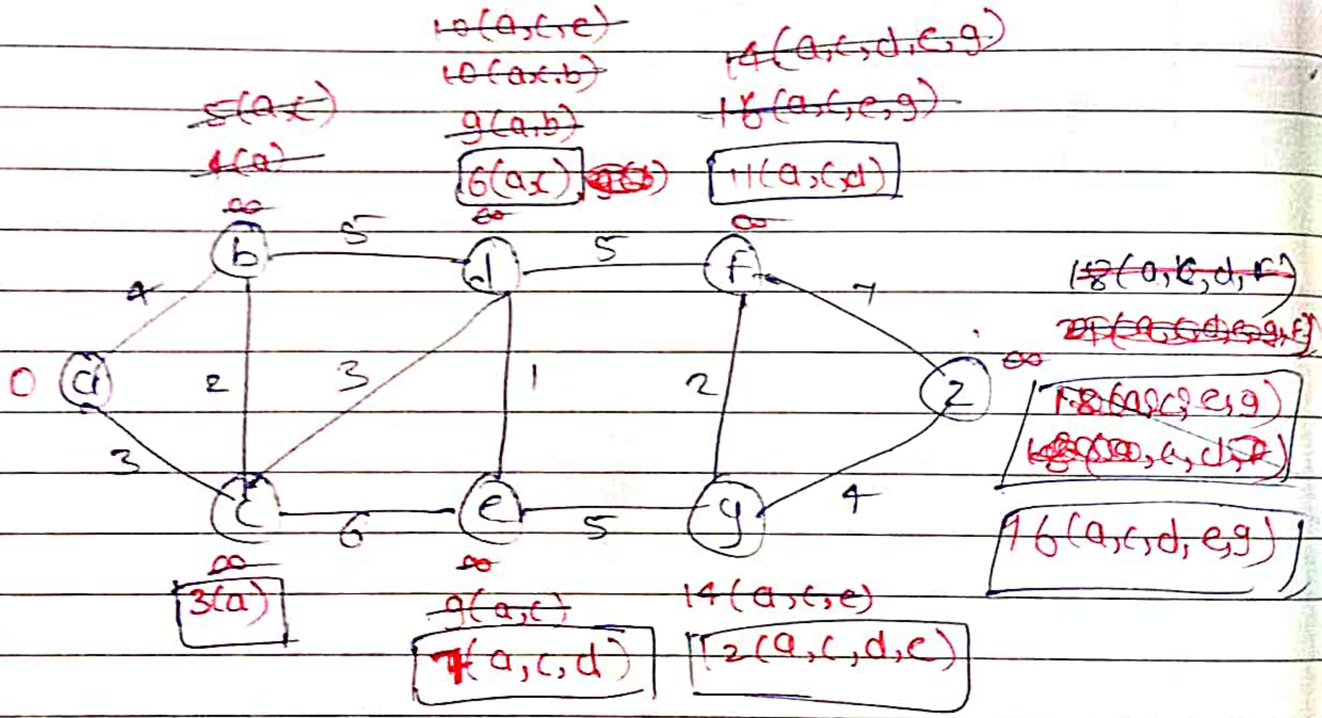
Single sourced shortest path.
(Dijkstra's Algorithm).

eg) 1)



2)

Imp:

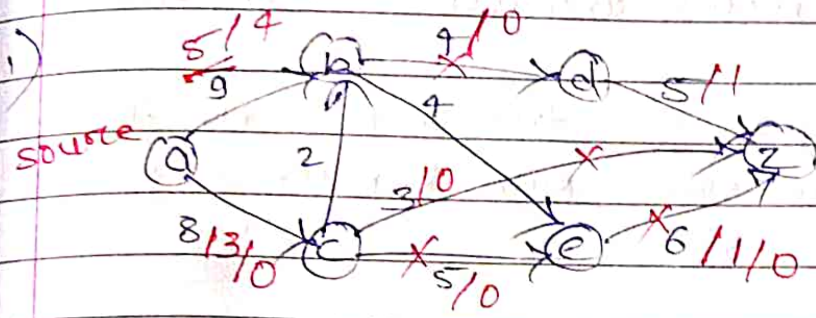


Shortest path = 16(a,c,d,e,g,z)

its length = 16.

Maximum flow using labeling algorithm.

1)



Source node / Start node : no incomming edges.

~~terminal node~~

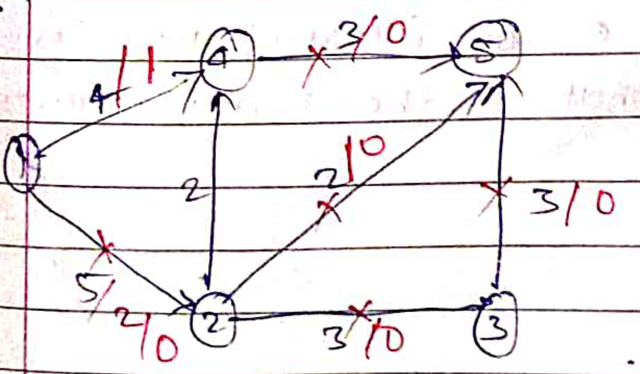
sink node / terminal node : no. outgoing edges.

path	max flow
a - b - d - z	4
a - c - e - z	5
a - c - z	3
a - b - e - z	1
	13 ~~units~~

∴ Max flow is 13 units.

2) Use labeling algorithm to find max flow for the given network below.

source = 1
sink = 5



path	max flow
1 - 4 - 5	3
1 - 2 - 3 - 5	3
1 - 2 - 5	2
	8

∴ Max flow is 8 units.

Unit IV

4. Trees.

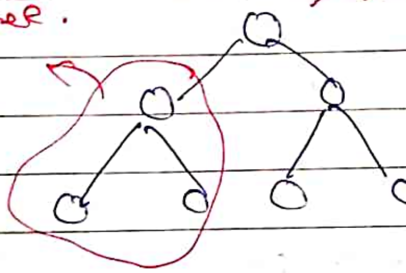
Trees.

→ A tree is a connected, undirected graph with no simple circuit

→ An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Root.

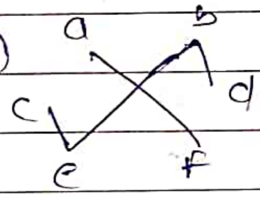
Sub tree.



leaf (last nodes).

eg ①

②



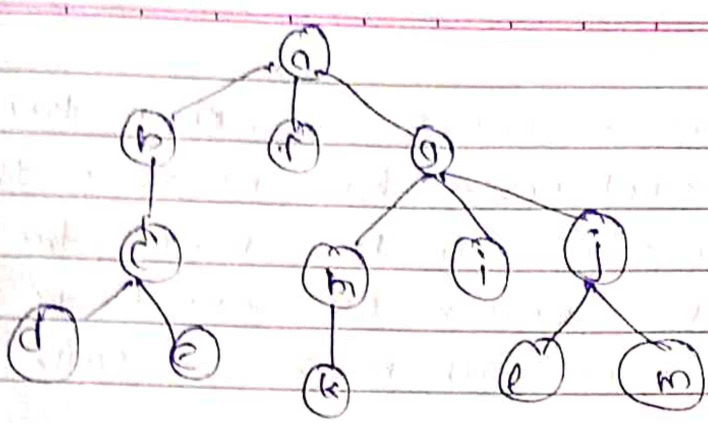
⇒ it is a tree. ⇒ not a tree.

Roted tree.

Rooted tree is a tree in which one vertex has been designated as a root and every edge is directed away from the root.

eg 1) In a rooted tree given below, find parent of 'c' the children of 'g', the sibling of 'h' and all descendence of 'b' and all ancestor of 'e' or internal vertices and all leaf nodes. What is the subtree rooted at 'g'.

⇒

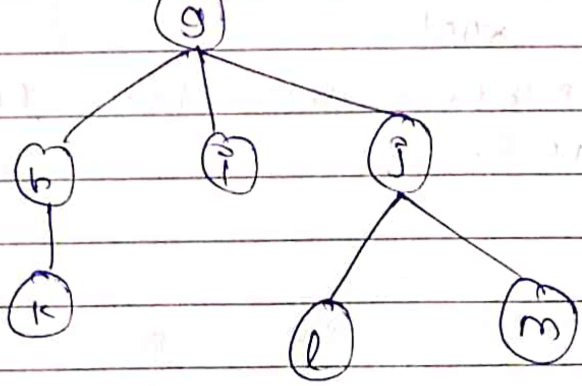


1. parent of c is b.
2. children of g = h, i, j.
3. sibling of h = i, j.
4. all descendent of b = c, d, e
5. all ancestor of e = c, b, a.
6. all internal vertices = ~~a~~ b, c ~~...~~ g, h, i, j.

(internal vertices are = all except leafs including root node.)

7. all leaf node = d, e, f, k, l, m.

8. subtree rooted at g.



Rooted tree.

→ A rooted tree is called m-ary tree if every internal vertex has no more than m children, the tree is a full m-ary tree if every internal vertex has exact m children and m-ary tree with $m=2$ is called binary tree.

Binary search tree.

→ Binary search tree is a binary tree in which each child of a vertex is designated as a right or left child.

→ No vertex has more than one right child or more than one left child.

→ And each vertex is labeled with a key (value).

→ Vertex / vertices are assigne keys so that the key of a vertex is both larger than the keys of all vertices if it's left subtree and smaller than keys of all vertices in a right subtree.

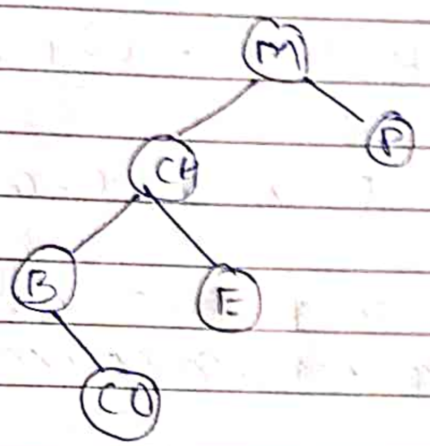
i.e. Right subtree must have greater or equal values (key) than root

and left subtree must have lesser values (key) than root.

eg. Construct binary search tree for the words MATH, PHY, CHEM, BIO, ENGLISH, COMP

⟹

Always 1st data is a root node.
Compare A/C to alphabetical order.

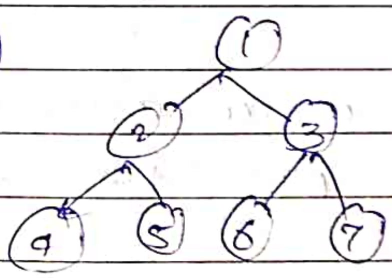


★ Tree Traversal.

→ 3 ways:

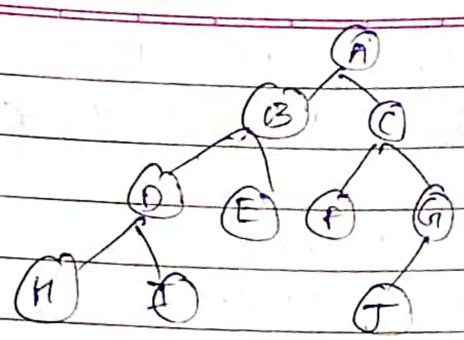
1) Preorder : Root left child Right child.
2) In order : Left Root Right
3) Post order : Left Right Root.

eg ①



⟹ ① Preorder : 1 - 2 - 4 - 5 - 3 - 6 - 7
② Inorder : 4 - 2 - 5 - 1 - 6 - 3 - 7
③ postorder : 4 - 5 - 2 - 6 - 7 - 3 - 1

(e)



⇒ Preorder : A - B - D - H - I - E - C - F - G - J

~~In o~~

In order : H - D - I - B - E - A - F - C - G - J.

Postorder : H - I - D - E - B - F - J - G - C - A

Level

→ The level of a vertex in a rooted tree is the length of the unique path from the root to this vertex.

→ The level of the root is defined to be zero.

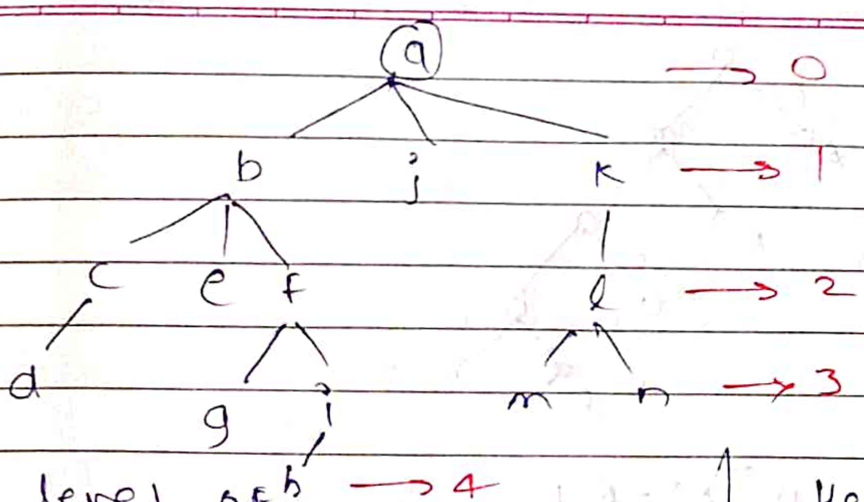
Height.

→ The height of the rooted tree is a maximum of the levels of vertices.

In other words, the height of the rooted tree is the length of the longest path from the root to any vertex.

eg. (1) Find the level of each vertex in the rooted tree shown in the following tree.

Also find the height of the tree.



→ level of

root a = 0

b, j, k = 1

c, e, f, l = 2

d, g, i, m, n = 3

h = 4

Height of tree is 4

It is a not Balenced tree.

NOTE :-

→ m = any integer no.

→ A rooted m-ary of height h is balenced if all leaf nodes are at level 'h' or 'h-1'

☆ Prefix code.

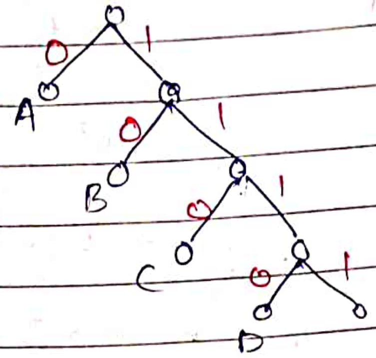
→ A text message can be converted into a sequence of 0 and 1 by replacing each character of the message with its code.

→ Prefix code can be represented by a binary tree data only at leaf node, code of each character can be found starting from the root of the tree and recording the path.

0 = left branch.

1 = Right branch.

eg



$\Rightarrow$ A = 0 $\Rightarrow$ length of (A) = 1

B = 1 0 $\Rightarrow$ l (B) = 2

C = 110 $\Rightarrow$ l (C) = 3

D = 1110 $\Rightarrow$ l (D) = 4

Imp.

☆ Huffman code.

eg

1) Use Huffman coding to encode the following symbol with the frequencies listed

A	0.08	E	0.20
B	0.10	F	0.35
C	0.12		
D	0.15		

What is the average no. of bits used to encode a character.

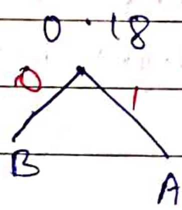
Steps:-

$\Rightarrow$ Given: (1) Arrange in ascending order.

0.08	0.10	0.12	0.15	0.20	0.35.
A	B	C	D	E	F

(2) consider two symbols with lowest freq^n. and merge them.

0.08 + 0.10 = 0.18. $\Rightarrow$

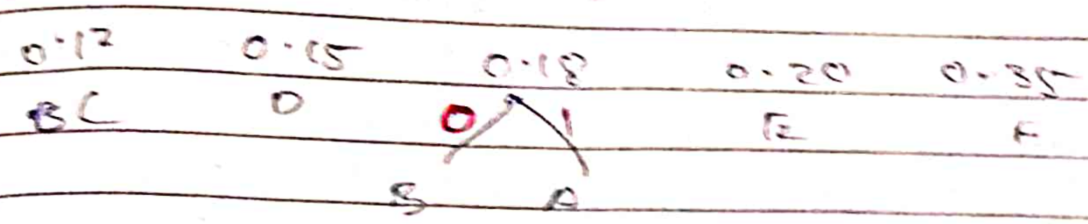


and form a tree.

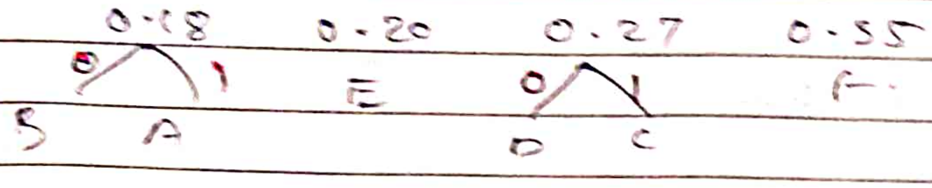
Right side = 1 = give data of lowest freq^n

left side = 0 : give data of not lowest freq^n.

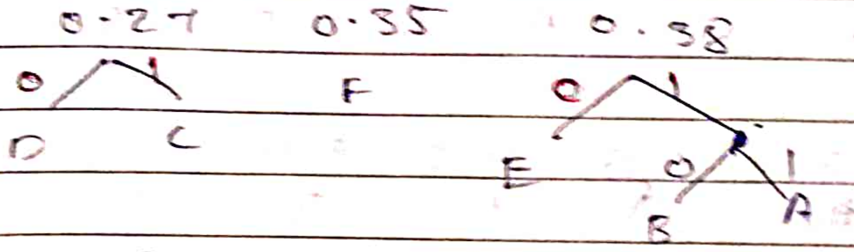
Step 1



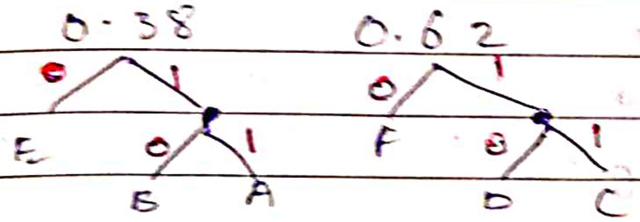
Step 2



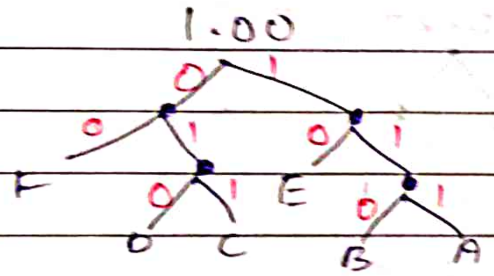
Step 3



Step 4



Step 5.



→ Prefix code for:

Cod	A = 111	$l(A) = 3$
	B = 110	$l(B) = 3$
	C = 011	$l(C) = 3$
	D = 010	$l(D) = 3$
	E = 10	$l(E) = 2$
	F = 00	$l(F) = 2$

→ Average no. of bits used to encode a symbol using Huffman coding is

length(A) f(A)

$= \underbrace{(3\times 0.05)}_{\text{for A}} + \underbrace{(3\times 0.10)}_{B} + \underbrace{(3\times 0.12)}_{C} + \underbrace{(3\times 0.15)}_{D} + \underbrace{(2\times 0.20)}_{E} + \underbrace{(2\times 0.35)}_{F}$

$= \underline{2.45}$ Ans.

Page No:
Date:

2) Use Huffman coding to encode with symbols with given frequencies. What is the avg no. of bits required to encode a symbol.

0.10	0.25	0.05	0.15	0.30	0.07	0.08
A	B	C	D	E	F	G

⇒ ~~Step1 0.05 0.10 0.~~
~~C~~

0.05	0.07	0.08	0.10	0.15	0.25	0.30
C	F	G	A	D	B	E

Step1

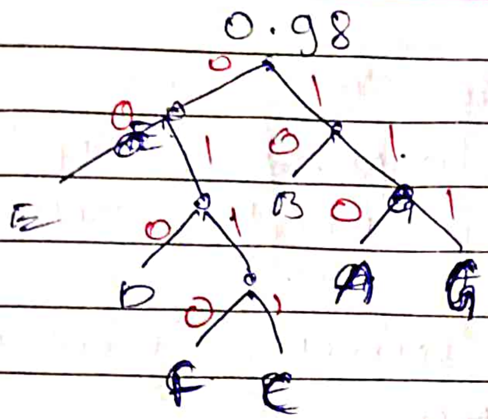
0.08 G   0.10 A   0.12 (F, C)   0.15 D   0.25 B   0.30 E

0.12 (C, F)   0.15 D   0.16 (A, G)   0.25 B   0.30 E

0.16 (G, A)   0.25 B   0.27 (D, (C, F))   0.30 E

0.27 (D, (F, C))   0.30 E   0.41 (B, (G, A))

0.41 (B, (G, A))   ~~0.57~~   0.57 (E, (D, (F, C)))



Prefix code for:

A = 110   l(A) = 3   E = 00   l(E) = 2

B = 10   l(B) = 2   F = 0110   l(F) = 4

C = 0111   l(C) = 4   G = 111   l(G) = 3

D = 010   l(D) = 3

→ Avg no. of bits used to encode a symbol using Huffman coding is:

$(3 \times 0.10) + (2 \times 0.25) + (4 \times 0.05) + (3 \times 0.15) + (2 \times 0.30) + (4 \times 0.07) + (3 \times 0.08)$

$= 2.57$

Ans.

Minimum Spanning Tree (MST)

→ Let G be a simple graph. A spanning tree of G is a subgraph of G i.e. tree containing every vertex of G.

→ A simple graph is connected if and only if it has a spanning tree.

→ MST:

A MST in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges. # if ~~in~~ Tree ⇒ n = no. of vertices then no. of vertices of ST = n - 1

→ Prim's Algorithm. "Method 1"

→ step 1:

To carry out prim's algorithm, begin by choosing any edge with smallest weight, putting it into the spanning tree.

→ step 2:

successively add to the tree edges of minimum weight that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree.

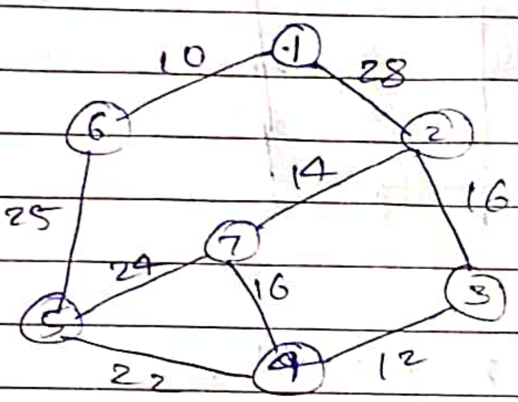
→ Step 3:

Stop when n-1 edges have been added where n = no. of vertices.

Q 1) Construct minimum spanning tree for the graph given below.

⇒

Here
No. of Edges of Tree



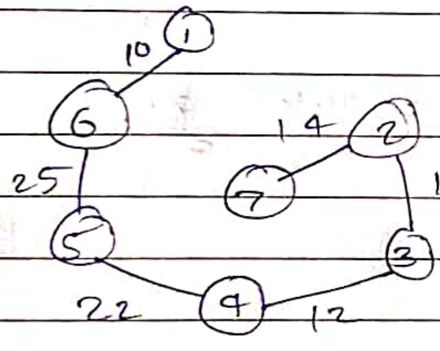
⇒

Here,

No. of ~~Edges~~ vertices of Tree = $|V| = 7$.

∴ No. of Edges of Spanning tree $= |V| - 1$

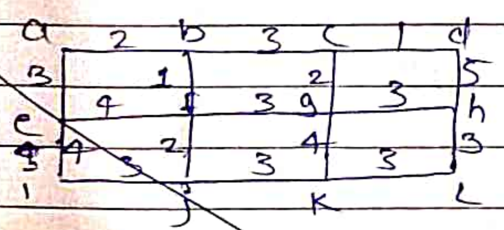
$= 7 - 1 = 6$.



Sr.No.	Edges	Cost
1	<1, 6>	10
2	<6, 5>	25
3	<5, 4>	22
4	<4, 3>	12
5	<3, 2>	16
6	<2, 7>	14

Total cost = 99

2)

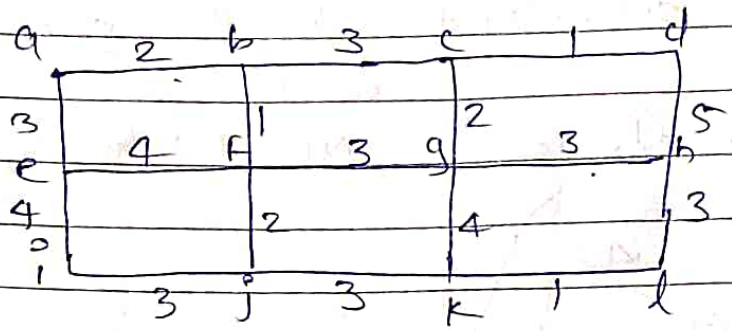


⇒

⇒ no. of vertices = 12

∴ no. of Edges = 11

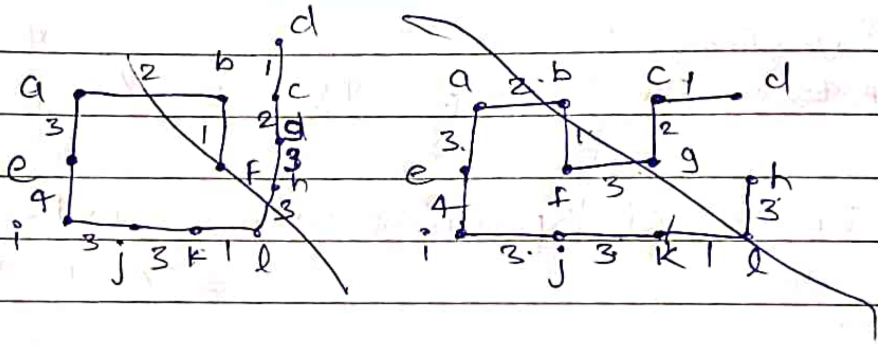
2)

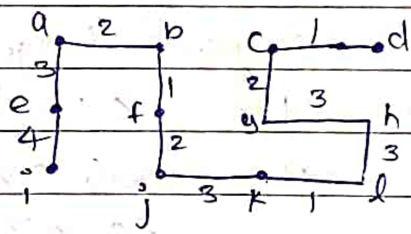


$\Rightarrow$ No. of vertices = 12

$\therefore$ Edges = ~~11~~ 12 − 1 = 11.

#





$\Rightarrow$ Total = 25.

Kruskal's algorithm

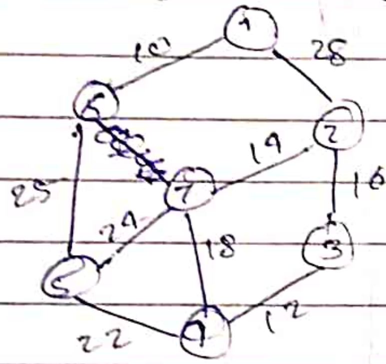
To carry out Kruskals algo : ~~choose an edge~~

→ Step 1:
Choose an edge with minimum weight.

→ Step 2:
successively add edges with minimum weight that do not form circuit with those edges already chosen.

→ Step 3:
Stop after n-1 edges have been selected.

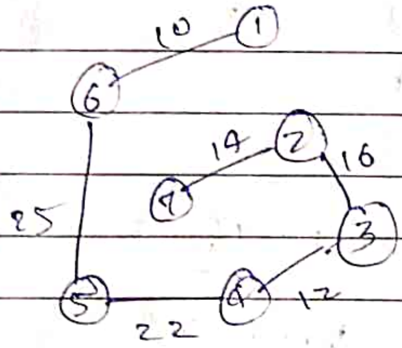
eg. Construct min. weight spanning tree using kruskal's algo.



⇒ $|V| = 7$ ∴ No. of edges $= 7-1 = 6$.

Sr.no.	Edge	Wt.
1)	<1-6>	10
2)	<3-4>	12
3)	7-2	14
4)	2-3	16
5)	5-4	22
6)	~~[illegible]~~	~~[illegible]~~
7)	5-6	25
	~~[illegible]~~	

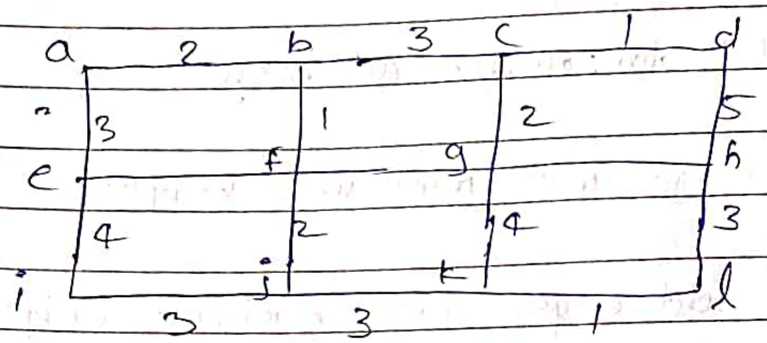
Total : 99



⇒ Edges may not be connected.

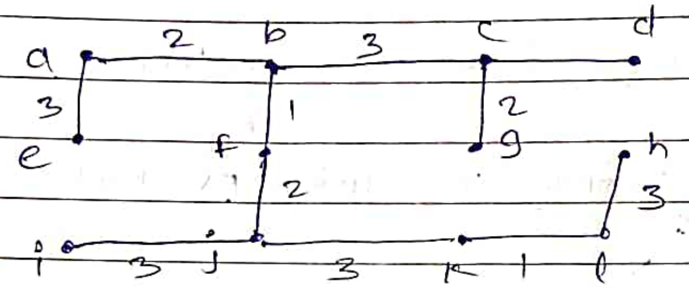
Page No.:
Date:

2) Find min spanning tree using Kruskal's algo.



#

⇒



Ans = 25 / 24

1)

c - d	1
k - l	1
b - f	1
c - g	2
f - j	2
a - b	2
h - l	3
j - k	3
i - j	3

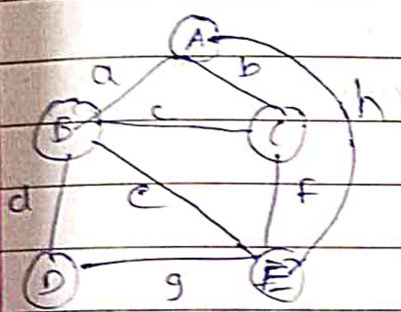
total = 18

Wrong ans

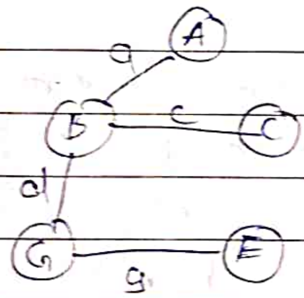
Fundamental circuit.

- Cutset partition all vertices into two disjoint sets.
- Cutset always contain only one branch and rest of the edges are chords.
- Branch is an edge associated with the spanning tree.
- Chord is an edge associated with the graph.
- When a circuit is formed by adding chord to the spanning tree is called fundamental circuit.

↳ always consider chord set.



Graph



Spanning tree of that graph.

Branches = {a, c, d, g}

Chords. = { b, e, f, h}

Edges	Fundamental
b	{a, b, c}
e	{d, e, g}
f	{c, d, g, h}
h	{a, d, g, h}

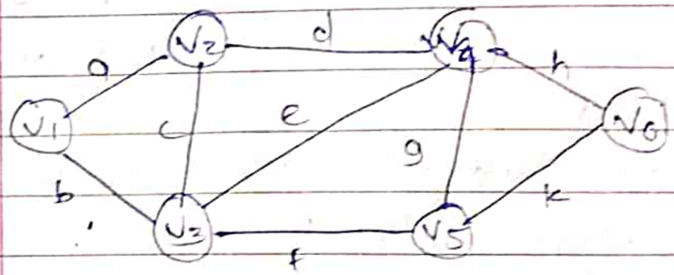
⇒ if b edge is attached then it will become circuit. or path {a, b, c}

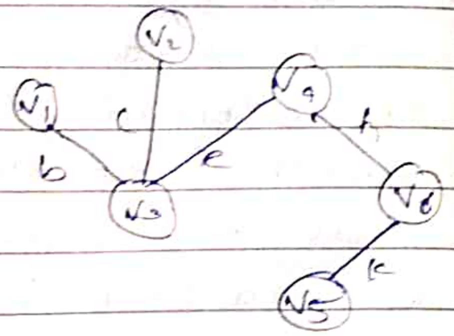
For Fundamental Circuit ⇒ Consider Chords
For Fundamental Cutset ⇒ Consider Branches.

↳ only contain single branch and rest are chords.

does not contain multiple branches.

eg Q. Find fundamental circuit and fundamental cutset for the graph g and tree given below.





⇒ (1) Branches = {b, c, e, h, k}

(2) Chords = {a, d, f, g}

(3)

Edges.	Fundamental ~~circ~~ circuit.
a	a, c, b
d	c, d, e
f	f, e, h, k
g	g, h, k

(4)

Edges	Fundamental cutset → Consider branches
b	~~{a, b}~~ {b, a}
c	{c, a, d} ⇒ When c is omitted, we can omit a and d to isolate V2 but we cannot omit b ∵ it is a branch (we can only omit 1 branch and other chords)
e	{e, d, f}
h	{h, g, f}
k	{k, f, g}

Here, we cannot omit k to isolate V6 ∵ it is 2nd branch. (1st branch is h)
∴ we have to omit g & f and thus V5 gets isolated.

{Isolating = dividing the graph into two parts.}

Unit V.

M T W T F S S
Page No.:
Date:
YOUVA

5. Algebraic Structure and Coding Theory.

Algebraic structure

~~A~~ If set A wrt any binary operator '$*$' satisfy closure property then it is called as a algebraic structure.

any binary operator

Properties: ~~$*$~~ ~~$\circ$~~ = any binary operation eg: (+, -, $*$, /)

1) Closure property. (Algebraic structure.)

→ Set A wrt operator '$*$' is said to satisfy closure property if $\forall a, b \in A$ then $a * b \in A$

~~then~~ if any operation is done on a and b ~~to~~ of set A ~~then~~ result must belong to set A. then it is called as closure property.

2) Associative property. (Semigroup.)

→ A binary operation '$*$' on the set A is said to be associative if $(a * b *) * c = a * (b * c)$ $\forall a, b, c \in A$.

3) Identity element (Monoid.)

→ An element 'e' belongs to set A is said to be an identity element for the operation $*$ if

$a * e = e * a = a$, $\forall a \in A$. a = any no.

e = identity element

4) Inverse property (Group.)

→ For each element $a \in A$ $\exists$ (there exists) an element $b \in A$ such that $a * b = e$, e = identity element.

$\boxed{a * a^{-} = e}$

5) ~~Comm~~ Commutative property (Abelian group.)

→ Binary operation '$*$' on the set A is said to be commutative if, $a * b = b * a$ $\forall a, b \in A$.

For ex: if we want to check ~~4th~~ Group i.e. 4th property then, we have to check all upper 3 properties then check 4th property. eg semigroup = Algebraic st. satisfying Associative property

eg Monoid = semigroup satisfying identity property.

Q1. Let $A = \{0, 1\}$ closed = closure property.

Is A closed under

① multiplication

② Addition.

⇒ For closure property: a

Here $a = 0$, $b = 1$

$a * b \in A$.

① For multiplication

$a \times b = 0 \times 1 = 0$

and $0 \in A$

∴ it satisfies closure prop.

② For Addition

$a + b = 0 + 1 = 1$

and $1 \in A$

∴ it satisfies closure prop.

⇒ ①

*	0	1
0	0	0
1	0	1

From this table it is observed that set A is closed under multiplication.

②

+	0	1
0	0	1
1	1	2

Here $2 \notin A$.

From this table, it is observed that 2 not belongs to set A. Hence, A is not closed under addition.

Page No.:
Date:

Q2. Consider G_1 be the set of all non-zero real numbers and $a * b = \frac{ab}{2}$.

Show that, $(G_1, *)$ is an Abelian group.

⇒ To check abelian group we have to check all the upper properties.

→ Step 1: Checking closure property.

As per prop., if $\forall\ a, b \in A$ then $a * b \in A$.

where here G_1 is the set of all non-zero real no's

Hence,

$\therefore\ a * b = \frac{ab}{2} \in G_1$

$\therefore$ It satisfies closure property.

→ Step 2: To check associative property.

$(a * b) * c = a * (b * c)\quad \forall\ a, b, c \in A$.

Here

G_1 = set of all non-zero real no's.

Hence

$a * b = \frac{ab}{2} \in G_1$

$\therefore$ it satisfies ~~closure~~ associat property.

→ ~~Step 3: To check Identity~~ $\quad a * b = \frac{ab}{2}$

$(a * b) * c = a * (b * c)$

LHS: $(a * b) * c$

$\therefore \left(\frac{ab}{2}\right) * (c)$

$\therefore \dfrac{\frac{ab}{2} * c}{2}$

LHS $= \frac{abc}{4} \in G_1$

Page No.:
Date:

RHS : $a * (b * c)$

$a * \left(\frac{bc}{2}\right) = \frac{a \; bc/2}{2}$

$RHS = \frac{abc}{4} \in G_1$

$\therefore LHS = RHS \in G_1$

Hence,
it satisfies associative property.

#

Step 3 : To check Identity element

$a * e = e * a = a$

$a * e = a$

$\therefore \frac{ae}{2} = a$

$\therefore e = 2 \Rightarrow 2 \in G_1$

Hence
$\therefore$ It satisfies identity property.

Step 4 : To check Inverse property

As per property, $a * b = e$

$a * a^{-1} = e$.

$a * a^{-1} = e$

$\frac{a a^{-1}}{2} = e$

$\frac{a a^{-1}}{2} = 2$

$\therefore a = \frac{4}{a}$ and $a = \frac{4}{a} \in G_1$

$\therefore$ It satisfies inverse property.

If $+n$ then it is a modulo (do %)
any no. is given
but only if $+$ is given then only do addition

→ Step 5: To check commutative property

$a * b = b * a$

$$\frac{ab}{2} = \frac{ba}{2}$$

LHS = RHS

∴ it satisfies commutative property.

→ ∵ It satisfies all 5 properties
Hence $(G, *)$ is an Abelian group.

Addition and Multiplication modulo. %

→ Consider Z be the set of integers and 'n' be any number, then Addition modulo of Z is denoted by $(Z, +_n)$.

→ And Multiplication Modulo n : $(Z, \times_n)$

1st 6 elements of Z_6.

Q. Show that, $Z_6, +_6$ is a group. ← Inverse prop. ∴ do addition then %.

6 Marks.

Here there is subscript, ∴ do %.

⇒ Steps.

i) Construct Addition Modulo table for $+_6$.

$(0+0)\%6$ $(0+1)\%6 = 1$

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

$(1+5)\%6 = 0$

$Z_6 = \{0, 1, 2, 3, 4, 5\}$

{ Closure property
$+_6$ ~~must belong to 26~~.

(2) To check closure property.
$\because$ All the entries in the table are the elements of $Z_6 = \{0, 1, 2, 3, 4, 5\}$
So, it satisfies closure property.

(3) To check associative property.
$(a * b) * c = a * (b * c)$ ← As per defination.
$\therefore (a +_6 b) +_6 c = a +_6 (b +_6 c)$ ← in terms of $+6$.
and $\forall a, b, c \in Z6$.
So.
from the table it is observed that it satisfies associative property.

(4) To ~~check~~ find Identity element.
$(a * e) = e * a = a$.
$\therefore a +_6 e = e +_6 a = a$.

From the table, 0 is the identity element.

(5) To check Inverse Property / to find inverse element.
~~$a * a^{-1} = e$~~
~~$a +_6 a^{-1} = e = 0$~~
~~$a \oplus a^{-1} = e$~~.
$a * a^{-1} = e$
$a +_6 a^{-1} = 0$
$0 +_6 a^{-1} = 0 \Rightarrow \therefore$ Inverse for $0 = 0$.
$1 +_6 a^{-1} = 0 \Rightarrow \therefore$ Inverse of 1 is = 5
$2 +_6 a^{-1} = 0 \Rightarrow \therefore$ Inverse of 2 is = 4
$3 +_6 a^{-1} = 0 \Rightarrow$ —11— 3 is = 3
$4 +_6 a^{-1} = 0 \Rightarrow$ —11— 4 is = 2
$5 +_6 a^{-1} = 0 \Rightarrow$ —11— 5 is = 1

Since,
it satisfies all 4 properties i.e.
closure, associative, identity & inverse
$\therefore$ It is a group.

Q. Show that, $G = \{1, 2, 3, 4, 5\}$ is not a group under Addition modulo 6. (i.e. $+_6$)

$\Rightarrow$ Constructing Addition modulo table for $(G, +_6)$.

(1) Construct $+_6$ table for set G.

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

(2) Checking closure property.
$\because$ From the table, it is observed that,
~~all elements~~
$0 \notin G$.
$\therefore$ It does not hold closure property
$\therefore$ It is not a Algebraic structure.
$\therefore$ It is not a group.

Q3. Consider $G = \{1, w, w^2\}$, $w^3 = 1$
Show that, $(G, \times)$ form a group.

$\Rightarrow$

(1) Construct table

$\times$	1	w	w^2
1	1	w	w^2
w	w	w^2	w^3
w^2	w^2	w^3	w^4

$\Rightarrow$ New table,

$\times$	1	w	w^2
1	1	w	w^2
w	w	w^2	1
w^2	w^2	1	w

#

$w^3 = 1$

$w^4 = w^3 \cdot w = 1 \times w = w.$

(2) To check closure prop.
from table it is observed that,
all elements $\in$ set G.
$\therefore$ It satisfies closure property.

(3) To check associativity.
$(a * b) * c = a * (b * c)$
$(a \times b) \times c = a \times (b \times c)$ — in terms of $\times$.
and $\forall a, b, c \in G$
$\therefore$ It satisfy associative prop.

(4) To ~~check~~ find Identity element.
$a * e = e * a = a$.
$a \times e = e \times a = a$
$\therefore$ $e = 1$, 1 is the identity element.

(5) To ~~ch.~~ find Inverse.
$a * a^{-1} = e$
$a \times a^{-1} = e = 1$
$1 \times a^{-1} = 1 \Rightarrow a^{-1} = 1$ for 1
$w \times a^{-1} = 1 \Rightarrow a^{-1} = w^2$ for w
$w^2 \times a^{-1} = 1 \Rightarrow a^{-1} = w$ for w^2.

and all $a^{-1} \in G$.
Since, it satisfies all 4 properties
∴ It is a group.

✩ Homomorphism and Normal sub-group.

any binary operation — another binary operation

Consider $(G, \circ)$ and $(G, *)$ are two groups then
f is mapping betⁿ two groups G and G' $f : G \to G'$
f = homomorphism betⁿ the two groups

any binary operation $(+, -, \times, /)$

$$\boxed{f(a \circ b) = f(a) * f(b)}$$

this operation belongs to group 1. — this operation belong / is of group 2.

if it holds this property,
then f is called as homomorphism betⁿ the groups.

Q1. Show that, $f : G \to G'$ defined by $f(x) = 2^x$
is a homomorphism, where $G = (R, +)$ and $G' = (R^+, \cdot)$ multiplication
R = set of all Real no's.
R^+ = set of all +ve Real no's. 2 - 3 Marks.

⟹

Property of homomorphism w.r.t two g
$f(a \circ b) = f(a) * f(b)$
∴ $f(a+b) = f(a) \times f(b)$ multiplication.
$\forall\ a, b \in G$.

$f(x) = 2^x$
∴ $f(a) = 2^a$, $f(b) = 2^b$.

$f(a+b) = 2^{a+b} = 2^a \cdot 2^b = f(a) \cdot f(b)$
LHS = RHS.
∴ LHS = RHS.
∴ It is a homomorphism.

Abelian group : Set with one operation $(A, *)$
RING : Set with 2 operations. $(A, +, *)$.

RING.

→ An Algebraic system $(A, +, *)$ is called a ring if the following conditions are satisfied:

1. $(A, +)$ is an Abelian group. ← commutative prop.
2. $(A, *)$ is a semi group. ← Associative prop.
3. The operation $*$ is distributive over the operation $+$. i.e.

$$a.(b+c) = a.b + a.c$$
$$(b+c).a = b.a + c.a$$

Q1. $Z_4 = \{0, 1, 2, 3\}$ w.r.t $+_4, \times_4$ is a Ring.

⇒ Here there is subscript ∴ do ~~addition and~~ then 1 %

Step 1: construct $+_4$ and $\times_4$ table.

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\times_4$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

* Ring with unity.

→ If Ring has an identity element wrt Multiplication then R is called Ring with unity

* Commutative Ring:

→ In Ring R if $a \cdot b = b \cdot a$ $\forall a, b \in R$ then R is called Commutative Ring.

ex: Show that $(Z, +, \cdot)$ is a commutative Ring with unity.

* Ring with and without zero divisor.

→ zero divisor:

If $a \cdot b = 0$ $\forall a, b \neq 0$ $\forall a, b \in R$ $(R, +, \cdot)$ are zero divisor and zero is a identity element wrt

→ Ring with zero divisor.

A ring R containing zero divisor is called a ring with zero divisor.

→ Ring without zero divisor

If $a \cdot b = 0$, $a = 0$ or $b = 0$ or both a and $b = 0$ and $\forall a, b \in R$, here R is called Ring without zero divisor.

M T W T F S S
Page No.:
Date:
YOUVA

can come theory on this topic.

Integral domain and Field.

Integral Domain	Field.
→ R - ring must satisfy: - commutative - Ring with unity. - without zero divisor → then it will be Integral Domain	→ R - ring must satisfy: - Commutative - Ring with unity. - Every non-zero element has multiplication inverse of R. then it will be a Field.

Q. Check, $(R, +, \cdot)$ is a integral domain or field.

$\Rightarrow$

Coding theory and Group code.

→ A sequence of letters from an alphabet is called as a word. And a code is a collection of words that are to be used to represent distinct message.

→ A block code is a code consisting of words that are of same length.

→ consider "A" denote the set of all binary sequences of length 'n'

EX-OR → '$\oplus$' be a Binary operation on set A such that for x & y in set A,

$x \oplus y$ is a sequence of length 'n' that has one's in those positions x & y differ, and has zero's in those positions x & y are same.

→ Ex-OR $\oplus$ ⇒ If input is same ⇒ o/p is zero.

→ consider 'x' be a word in set A, we define weight of x '~~W(x)~~' denoted by $w(x)$ is the no. of one's in x.

eg: $x = 0\,1\,0\,1\,0$ ⇒ $w(x)$ = no. of ones = 2.

$y = 0\,0\,0\,1\,1$ $w(y) = 2$

EX-OR

$x \oplus y = 0\,1\,0\,0\,1$ $w(x \oplus y)$ = Hamming distance = 2.

* Hamming distance

Hamming distance is a dist. betn x & y denoted by $d(x,y)$ and it is the weight of $x \oplus y$. ~~w~~

i.e. $\boxed{d(x,y) = w(x \oplus y)}$

Page No.:
Date:

Q. Find hamming distance betⁿ
$x = 1\ 1\ 1\ 0\ 0\ 0\ 0$
$y = 1\ 0\ 0\ 1\ 1\ 0\ 0$

$x \oplus y = 0\ 1\ 1\ 1\ 1\ 0\ 0$
$d(x, y) = w(x \oplus y) = 4$

NOTE:

The distance betⁿ two words is exact the no. of positions at which they differ.

Parity check code

→ The following encoding fun $e: B^m \rightarrow B^{m+1}$ is called parity $(m, m+1)$ check code.

→ Suppose, $b = b_1, b_2, \ldots b_m \in B^m$ (Binary sequence)
then encoding funⁿ $e(b) = b_1, b_2, \ldots b_m, b_{m+1}$
where,

$$b_{m+1} = \begin{cases} 0 & \text{if } |b| \text{ is even} \\ 1 & \text{if } |b| \text{ is odd} \end{cases}$$

and $|b|$ = weight of b.
i.e. $|b| = w(b)$.

eg', $b = 0010$
$e(b) = 0010?$ ⇒ ∵ $|b| = 1$ = odd.
∴ Add 1

∴ $e(b) = 0010\underline{1}$

→ e: $B^m \longrightarrow B^{m+1}$ B = no. of bits.

$e(000) \longrightarrow 0000$
$e(001) \longrightarrow 0011$
$e(010) \longrightarrow 0101$
$e(011) \longrightarrow 0110$
$e(100) \longrightarrow 1001$
$e(101) \longrightarrow 1010$
$e(110) \longrightarrow 1100$
$e(111) \longrightarrow 1111$

α

word = binary sequence.
weight = no. of one's present in that word.

Page No.
Date

no. of bits.

eg' 1) consider (3,4) parity check code.
for each of the received word, determine whether an error will be ~~detected~~. detected.

a) 0010
b) 1001

given o/p. or received words

⟹

a) 0010 ⟹ Yes, error detected.
since, weight of 0010 is = 1 which is odd
∴ error is detected.

b) 1001 ⟹ NO Error.
since, weight of 1001 is = 2 which is even
∴ error is not present.

6 bit i/p 7 bit o/p.

2) consider (6,7) parity check code

a) 0110000
b) 1110000

⟹ a) 0110000 ⟹ NO error.
∵ weight of given word 011000 is 1 which is even ∴ no error.

b) 1110000 ⟹ Yes, error.
∵ $w(\text{word}) = 3$ which is odd
∴ error is detected.

Q1. Consider (2,6) encoding fun e

$e(00) = 000000$
$e(10) = 101010$
$e(01) = 011110$
$e(11) = 111000$

0	0	0	0	0	0
1	0	1	0	1	0
0	1	1	1	1	0
1	1	1	0	0	0

(1) Find minimum distance of e.
(2) How many errors will e detect?

⇒ (1). $d(e(00), e(10)) = \cancel{4}\ 3$
$d(e(00), e(01)) = 4$
$d(e(00), e(11)) = 3$
$d(e(10), e(01)) = 3$
$d(e(10), e(11)) = 2$
$d(e(01), e(11)) = 3$

∴ The minimum distance is 2.

(2) ~~e(00)~~ A code will detect 'k' or few errors if and only if its minimum distance is at least $k+1$.
since, here the min. distance is 2 so we have ~~so the code will~~ $2 \geq k+1 \Rightarrow \therefore k \leq 1$
so the code will detect one or few errors.