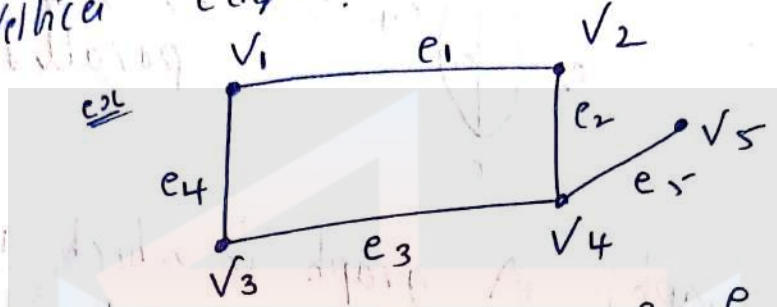
Unit 1 Preliminaries, Connected graph & shortest path, Trees

→ **Graph:** A graph G is a pair of set of

(V, E)

↑ ↑

vertices edges.



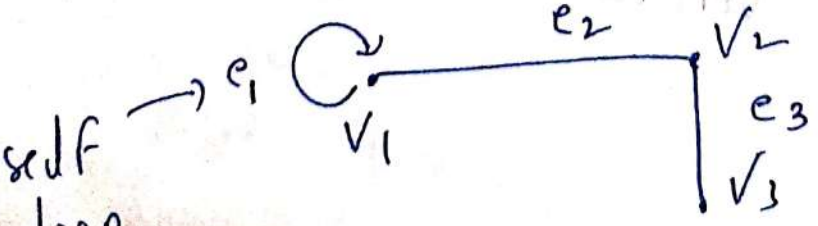
5 edges => { e1, e2, e3, e4, e5 }

5 vertices -> { V1, V2, V3, V4, V5 }

→ **Null Graph:** A graph in which number of edges = 0

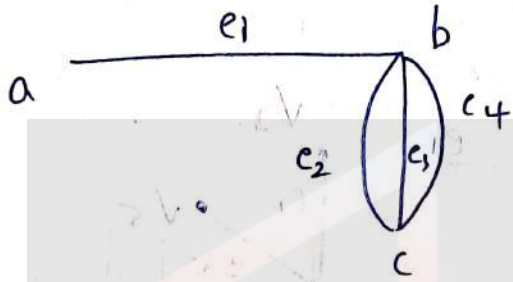
4 vertices & 0 edges

→ **self loop:** An edge joining a vertex to itself is called self loop



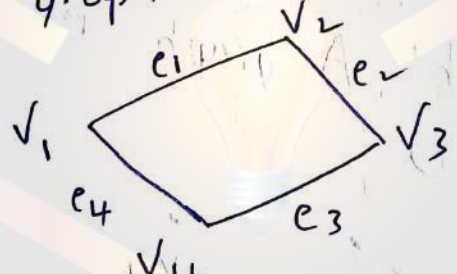
→ Parallel (or) multiple edges:-

In a graph it may be possible to have more than one edge with a single pair of vertices such edges are called parallel edges

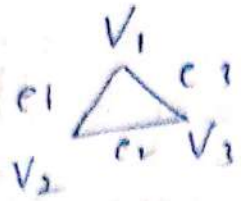


e_2, e_3, e_4 are parallel edges

→ Simple Graph:- A graph in which it contains neither self loops nor parallel edges is called Simple graph.



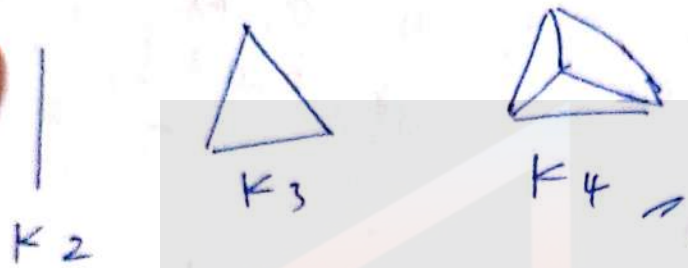
→ Complete graph:- A simple graph in which there is exactly one edge b/n each pair of distinct vertices is called complete graph.



⇒ For vertices 50

$^{50}C_2$ - 1225 ← total edges

∴ For n vertices $^nC_2$ edges will be their in complete graph

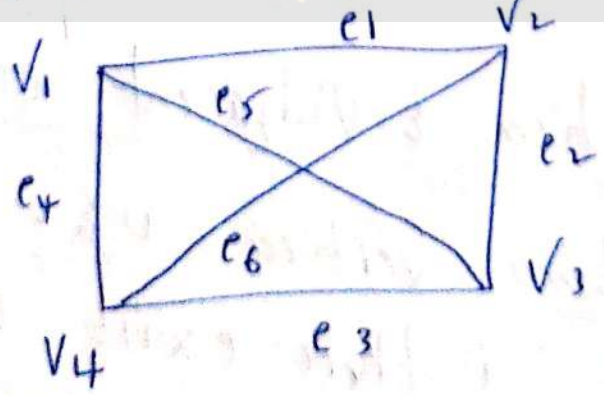


→ ~~Mouth graph:~~

→ Order & Size of graph:

The no. of vertices in a graph G is called order

The no. of edges in a graph is called size.

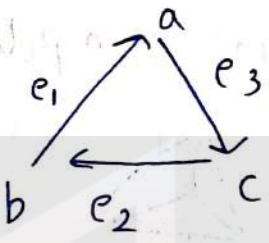


order = 4

size = 6

→ Directed Graph:

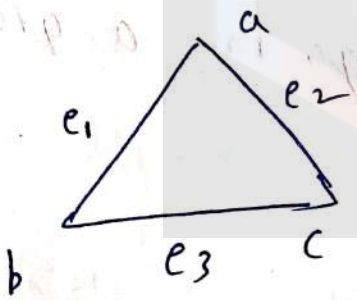
The graph in which the elements of the edge are ordered pair of vertices is called directed graph or digraph.



(a, c)
(c, b)
(b, a)

→ Non Directed Graph:

A graph which the elements of the edge set are unordered pair of vertices is called anon directed graph.



$e_1 = \{a, b\}, \{b, a\}$

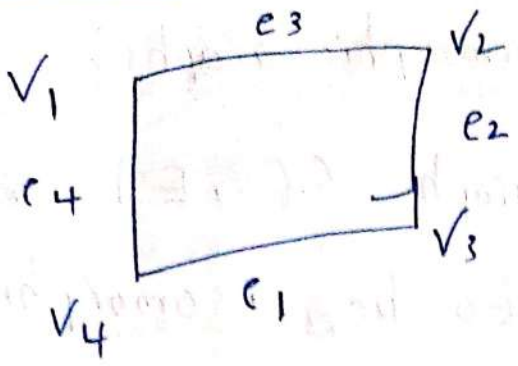
$e_2 = \{a, c\}, \{c, a\}$

$e_3 = \{b, c\}, \{c, b\}$

→ Adjacent vertices & Adjacent edges

→ two vertices u & v are said to be adjacent if there exist an edge

→ If two edges have a common vertex then they are called adjacent edge.



V_1 & V_2, V_2 & V_3, V_3 & V_4, V_4 & V_1 are adjacent vertices

e_1 e_2, e_2 e_3, e_3 e_4, e_4 e_1 are adjacent edges.

→ Finite & Infinite graph:

A graph is a finite if both its vertices set & the edge set are finite. Otherwise infinite.

→ weighted graph: A graph in which weights are assigned to every edge is called a weighted graph

1, 2, 3, 4 are weights of the edges.

→ **Isomorphism / Isomorphic graphs:**

→ Two graphs $G(V, E)$ & $H(W, F)$ are said to be isomorphic if their exist bijections $f: V \to W$ & $g: E \to F$ such that an edge e joins u & v in G if and only if the edge $g(e)$ joins $f(u)$ and $f(v)$ in H.

→ If G & H are isomorphic, we write $G = H$, and the pair (f, g) is called isomorphism b/n G & H.

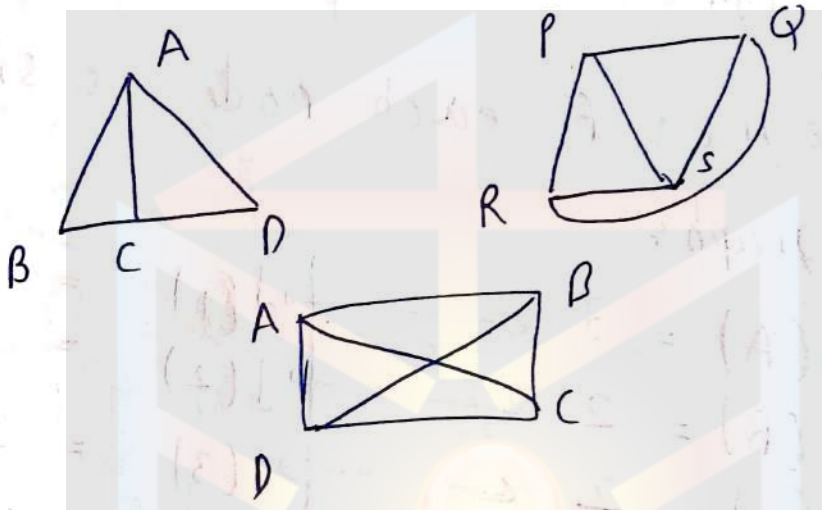
(or) (a simple method)

→ Two graphs are said to be isomarphic if they are perhaps the same graphs just drawn differently with different names i.e, they have identical behaviour for any graph theoretic properly.

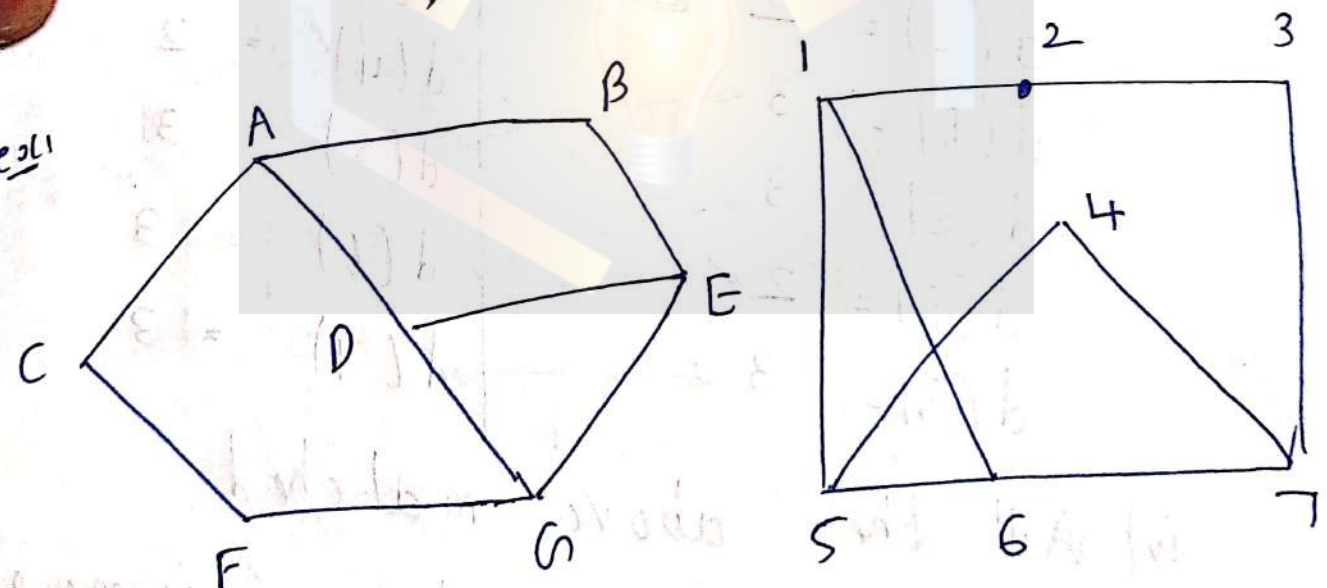
Properties

→ no. of vertices are equal
→ no. of edges are equal.
→ degree of each node is similar in G_1 & G_2
→ Individual G_1 cycle length = G_2 cycle length.

ex1



ex1



* Properties

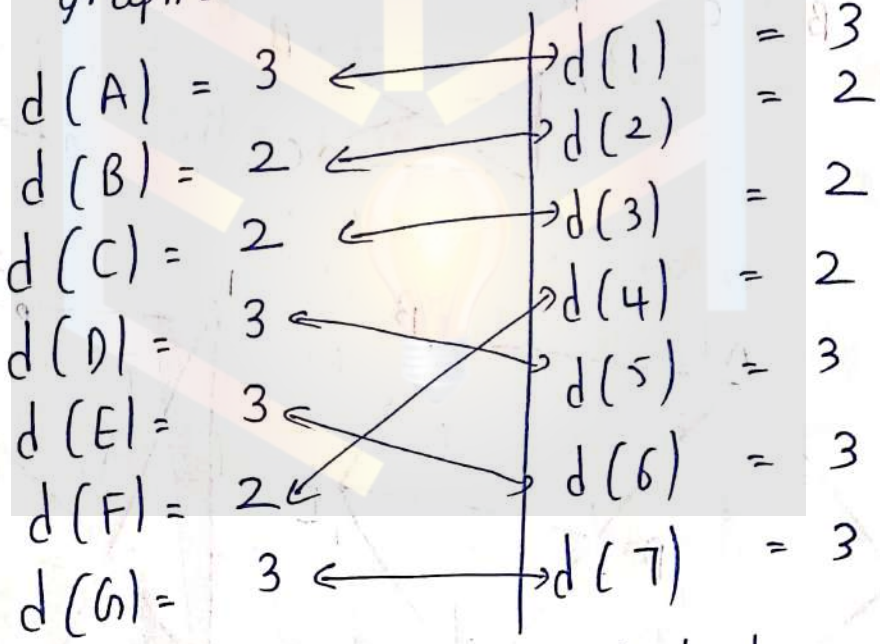
i) no. of vertices are equal

no. of vertices = 7	no. of vertices = 7 ✓

ii) no. of edges are equal.

no. of edges = 9	no. of edges = 9

iii) Degree of each node is similar in both graphs

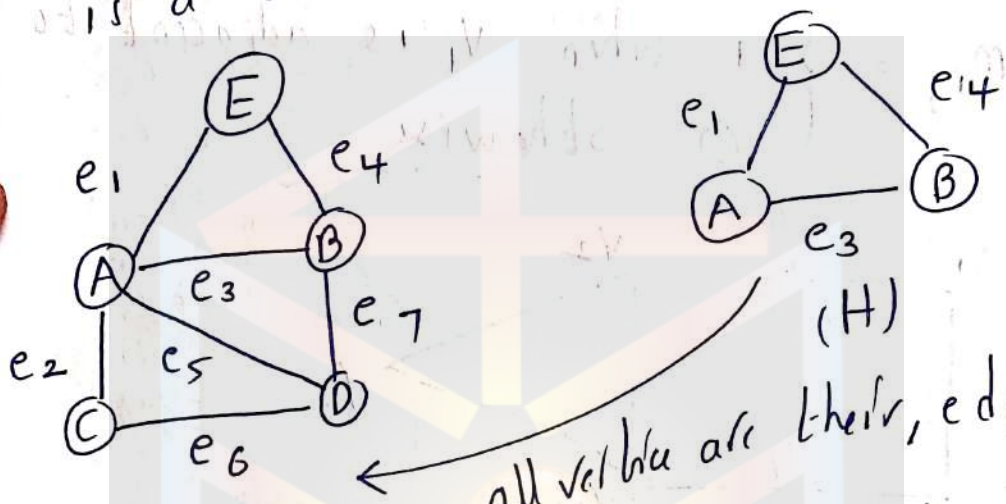


iv) All the above matched

Hence both graphs are isomorphic.

→ Subgraph: Suppose there are two graph, $H = (V_2, E_2)$ $G = (V_1, E_1)$

H is a subgraph of G if set of vertices of graph H is a subset of graph G. Set of edges of graph H is a subset of graph G.



all vertex are their, edges.

→ every graph is a subgraph of itself.

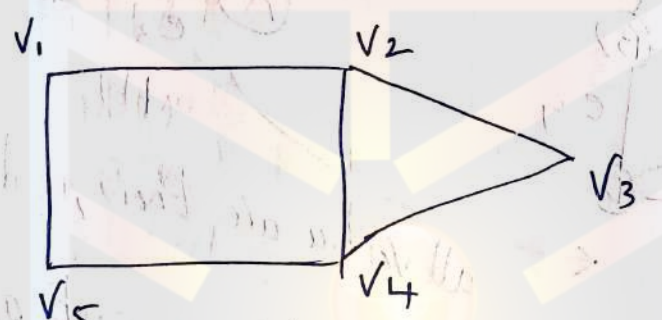
→ every graph with n vertices is subgraph of complete graph K_n

→ Adjacency matrix of a Graph (representation)

Let $G(V, E)$ be a simple graph with n vertices ordered from v_1 to v_n then the adjacency matrix $A = [a_{ij}]_{n \times m}$ of G is an $n \times m$ matrix.

$$m_{ij} = \begin{cases} 1 & \text{when } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Eg:



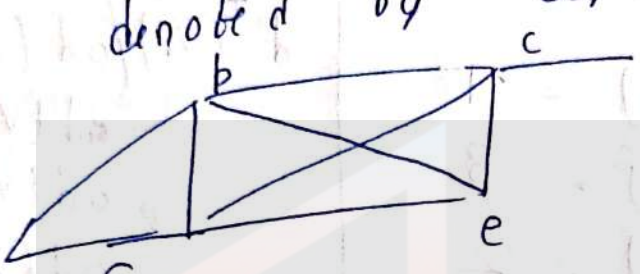
(G)

$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

→ Degree of a vertex

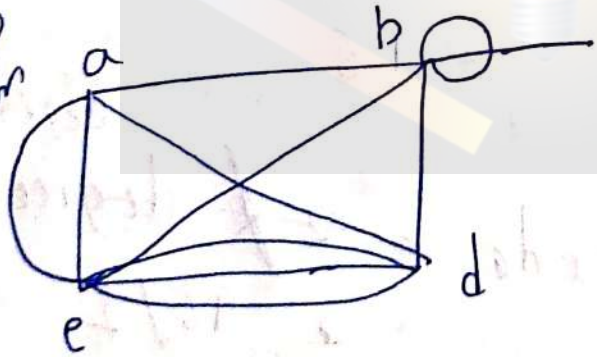
degree 0 = isolated
degree 1 = pendent

The degree of a vertex in an undirected graph is the number of edges incident with it except that a loop denoted by deg ()



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

For loop consider 2 (undirected



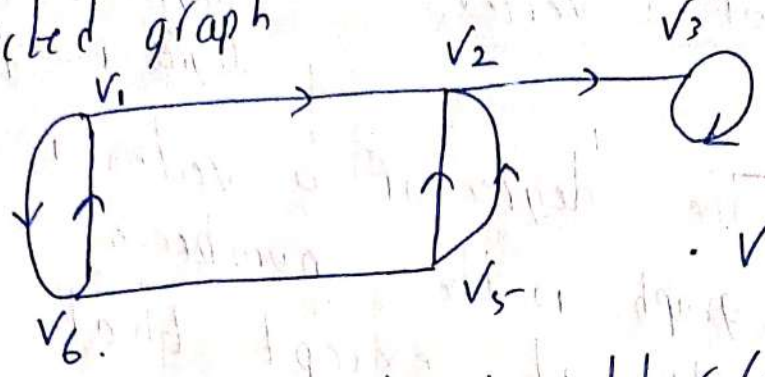
Note loop is consider degree 2

deg (a) = 4
deg (b) = 6
deg (c) = 1
deg (d) = 5
deg (e) = 6
22

Total degree = 2 × no. of edge

$\frac{22}{2}$ = no. of edge = 11

in directed graph



in degree (incoming edge)	out-degree (out-going edge)
$d^-(v_1)/id(v_1) = 1$	$d^+(v_1)/od(v_1) = 2$
$id(v_2) = 3$	$od(v_2) = 1$
$id(v_3) = 2$	$od(v_3) = 1$
$id(v_4) = 0$	$od(v_4) = 0$
$id(v_5) = 1$	$od(v_5) = 2$
$id(v_6) = 1$	$od(v_6) = 2$
8	8

$$id + od = 8 + 8$$
$$= 16$$

degree / 2

We know

$$\text{no. of edges} = \cancel{2} \cancel{\text{degree}}$$
$$= 2 \quad 16/2$$
$$= 8$$

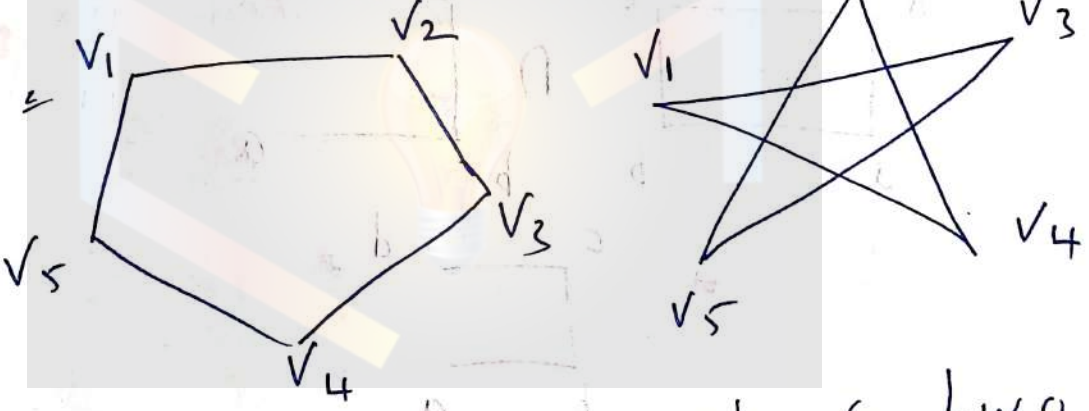
degree = 2 × no. of edge → hand shaking propelty

→ Operations on graph :

→ ~~It is a simple graph G~~

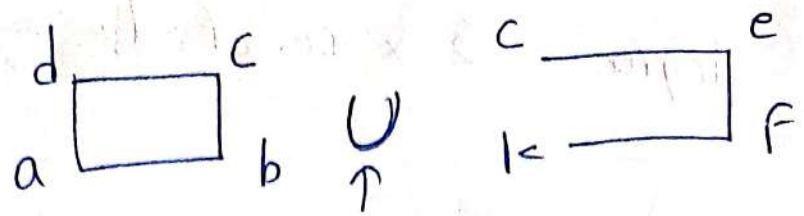
i) Complement of a Graph :

Complement of a Graph G. ~~if there is~~ marks an edge b/n two vertices u & v. if & only if their is no edge b/n u & v in G

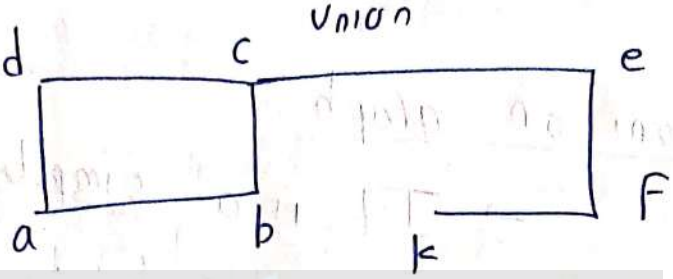


ii) Union : If G & G' are two graphs ~~are obt~~ then union of these graph is obtained by taking the union of vertex set & edge set

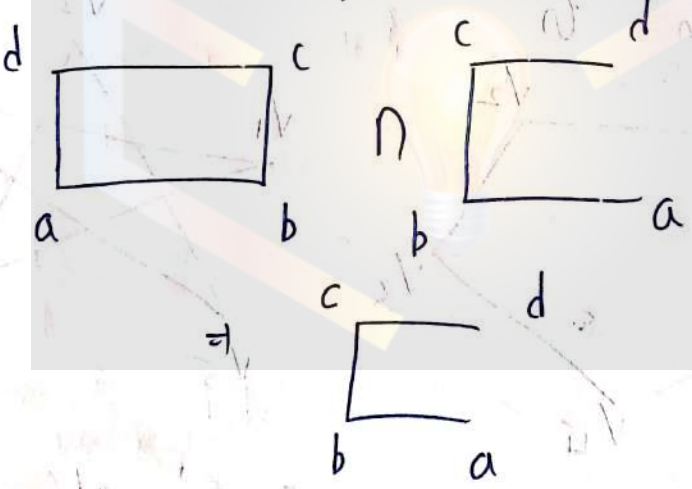
ex



⇒

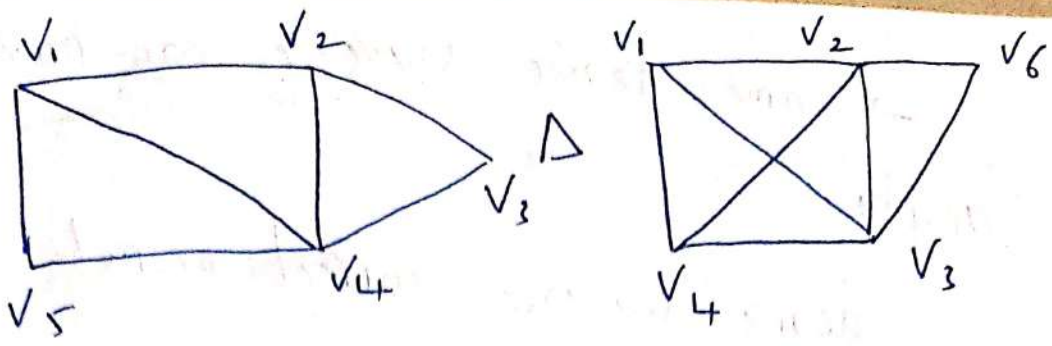


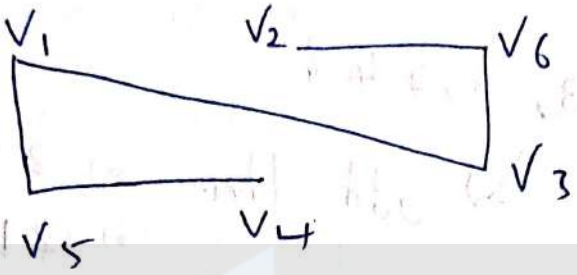
iii) Intersection: Let G & G' are two graph then the intersection of these graph are obtained by taking intersection of. vertex set & edges set



iv) Ring sum of graph: $G_1 \Delta G_2$

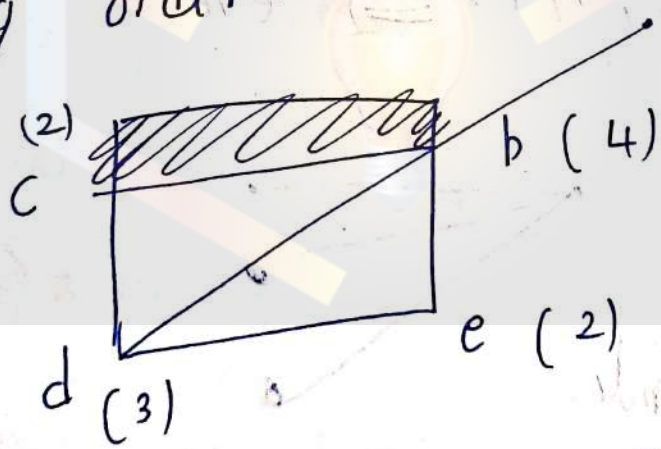
$$G_1 \Delta G_2 = (V_1 \cap V_2, E_1 \Delta E_2)$$





v) deletion: deleting a particular vertex.

→ Degree Sequence: The arrangement of degree of all vertices of a graph G either in non increasing or non decreasing order.



{4, 3, 2, 2, 1} → Degree seq

(or)

{1, 2, 2, 3, 4}

→ same degree sequence can have multiple graphs.

degree sequence correct or not.

{2, 3, 3, 5, 4 4}

add them ⇒ 2+3+3+5+4+4
⇒ 21 ✗

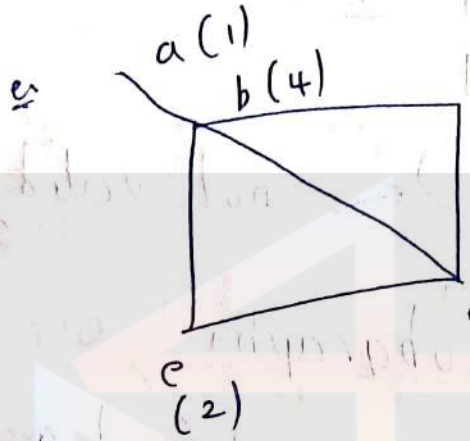
wrong degree seqv. as degree = 2 * edge.

& graph not possible.

{2, 5, 4, 3, 4} ⇒ even ✓

5 is not even possible.

→ Havel Hakmi theorem the is used to identify whether a given degree sequence of graph is true/ possible or false/not possible.



⇒ degree sequence = 4, 3, 2, 2, 1

remove that vertex

4 3 2 2 1
2 1 1 0
1 1
0 0 0

hence it is valid sequence.

(5, 4, 4, 3, 2)

1 1 1 1 (1) ← not possible

even if we get -ve degree it is invalid.

6 6 6 3 3 2 2
− 1 1 1 1 1 1

5 5 5 2 2 1 2
− 1 1 1 1 1

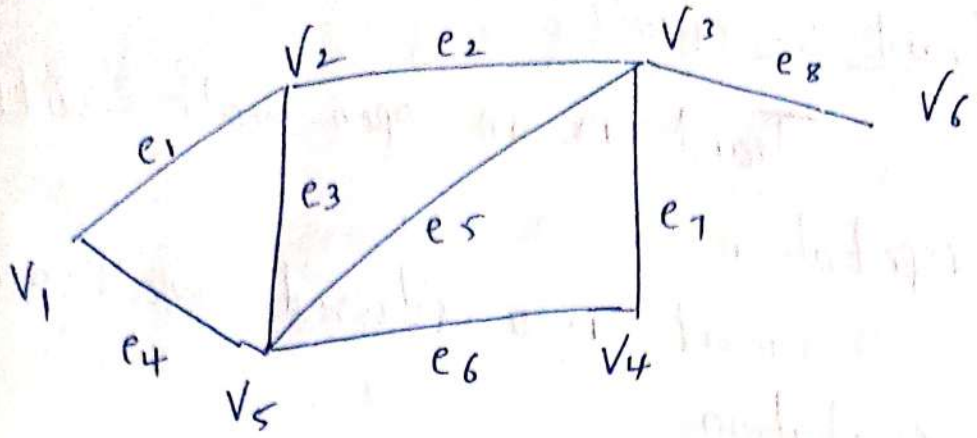
4 4 1 1 0 2
− 1 1 1 1

3 0 0 (−1) ← not valid //

→ 5 important subgraphs

Walk :- A walk is defined as a finite alternating sequence of vertices & edges begining & ending with vertices such that ~~each edge is incident with the vertices preceding & following it.~~

Note → no edge appear more than once in a walk (vertex however may appear more than once.



walk or not

⇒ $v_1\ e_1\ v_2\ e_2\ v_3\ e_8\ v_6$.

no edge repeated hence walk

⇒ $v_1\ e_4\ \underline{v_5}\ e_3\ v_2\ e_2\ v_3\ e_5\ \underline{v_5}\ e_6\ v_4$

no edge repeated hence walk.

→ length of walk is the no. of edges in the walk.

→ walk begins & end with same vertex is called closed walk

→ else open walk

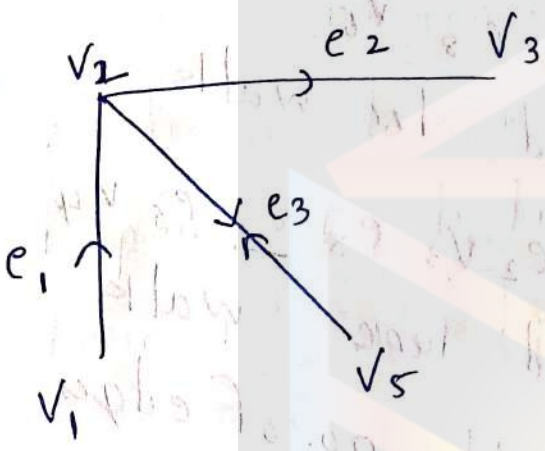
$\underline{v_1}\ e_4\ v_5\ e_3\ v_2\ e_1\ \underline{v_1}$ closed

$v_1\ e_4\ v_5\ e_3\ v_2$ open.

II) Traid & circuit :-

Traid is an open walk with no edge repetation

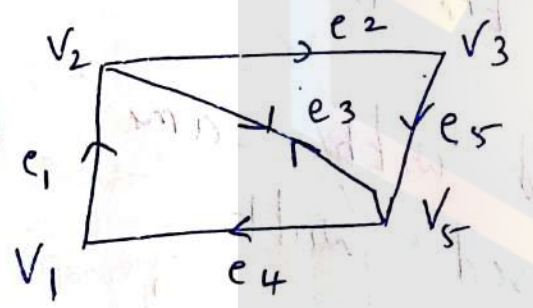
Circuit is a closed walk with no edge repetation.



$v_1 e_1 v_2 e_3 v_5 e_3 v_2 e_3 v_2$

open walk ✓

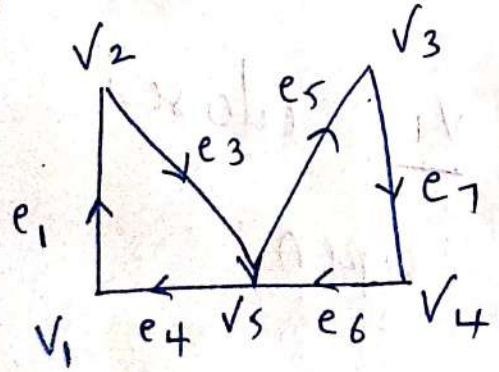
Traid ✗ circuit ✗



$v_1 e v_2 e_3 v_5 e_3 v_2 e_2 v_3 e_5 v_5 e_4 v_1$

closed walk ✓

circuit ✗ Trial ✗



$v_1 e_1 v_2 e_3 v_5 e_5 v_3 e_7 v_4 e_6 v_5 e_4 v_1$

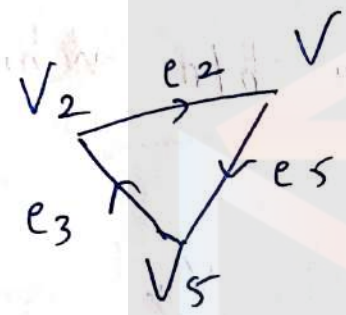
closed walk ✓

Circuit ✓

III) Path & Cycle :-

A trail in which no vertex repeated more than once is called path

A circuit in which terminal vertex as an internal vertex & nointernal vertex is repeated is called cycle.

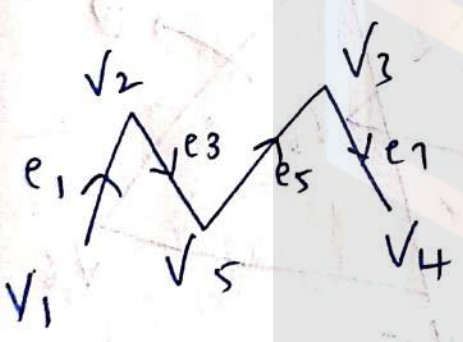


$V_2 e_2 V_3 e_5 V_5 e_3 V_2$

closed walk ✓

edge rep ✗ vertex rep ✗

cycle - ✓.



$V_1 e_1 V_2 e_3 V_5 e_5 V_3 e_7 V_4$

open ✓

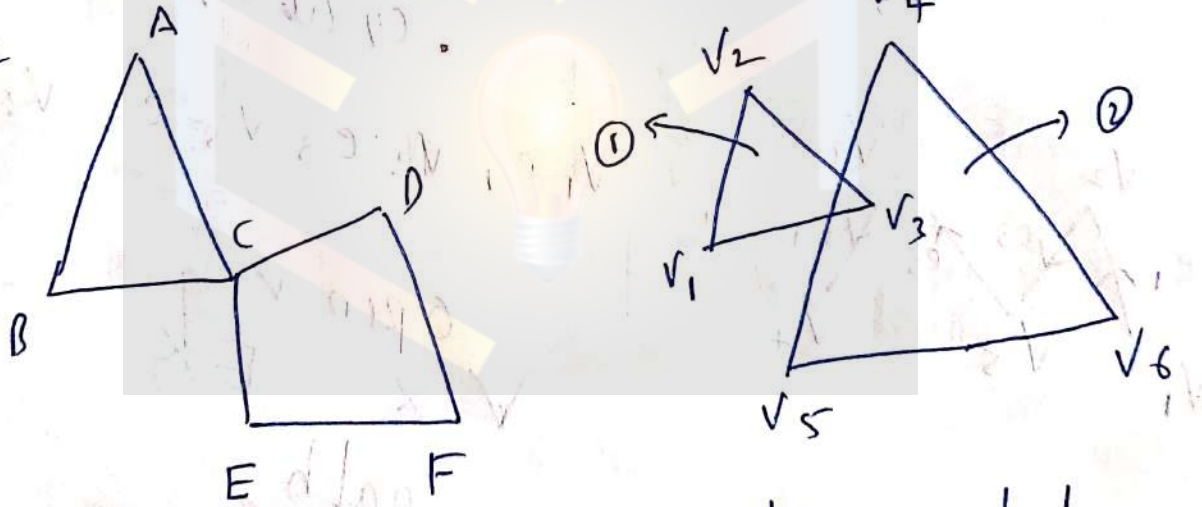
✓ ✗ EX

path =.

→ Connected & Disconnected graph:

Connected: A Graph G is said to be connected if their exist atleast 1 path b/n every distinct vertices in G

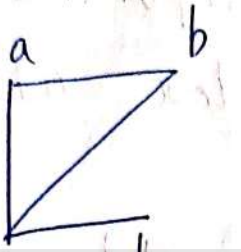
disconnected:- A Graph G is said to be disconnected if their has atleast 1 pair of distinct vertices b/n which their is no path.



connected graph

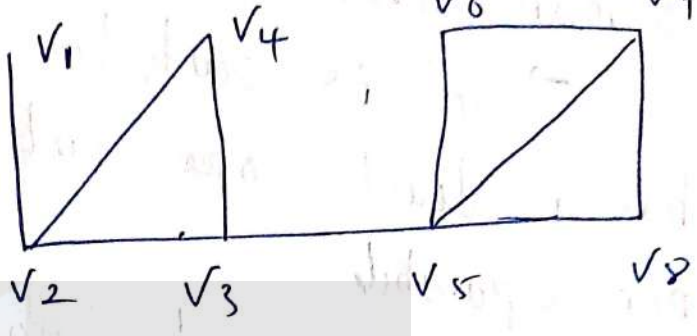
disconnected graph with 2 component

→ Cut vertex :- It is a vertex by which if we remove that vertex then the graph will be disconnected



removal of c

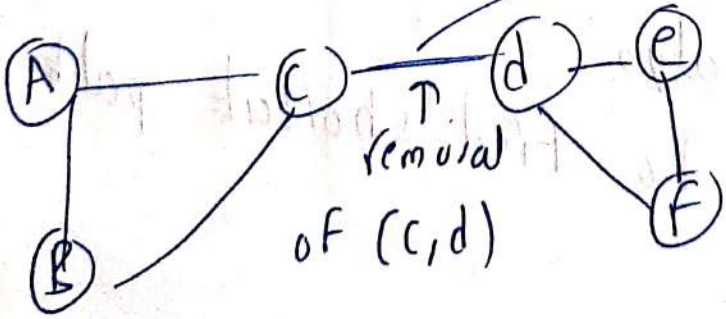
2 component (.d)



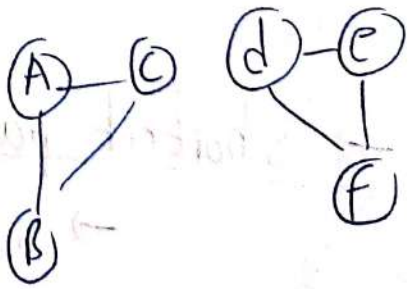
By removing v_2 or v_3 or v_5 we can make it disconnected

hence v_2, v_3, v_5 are cut vertices

→ Cut edge : It is an edge by which if we remove that edge then graph will be disconnected → also called bridge



⇒

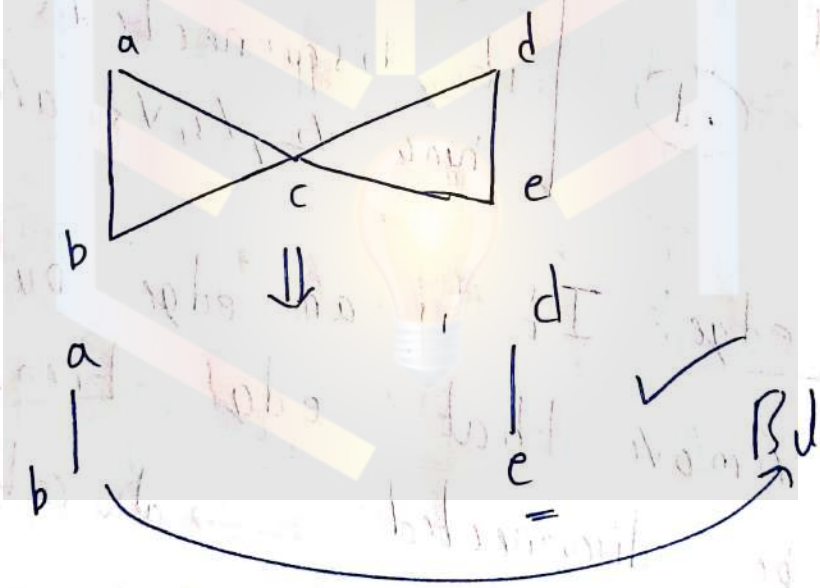


→ Blocks:

→ Let G be a connected graph on three or more vertices.

→ G is said to be separable if it has a least one cut vertex. otherwise non separable.

→ A maximal nonseparable connected subgraph of G is called block of G



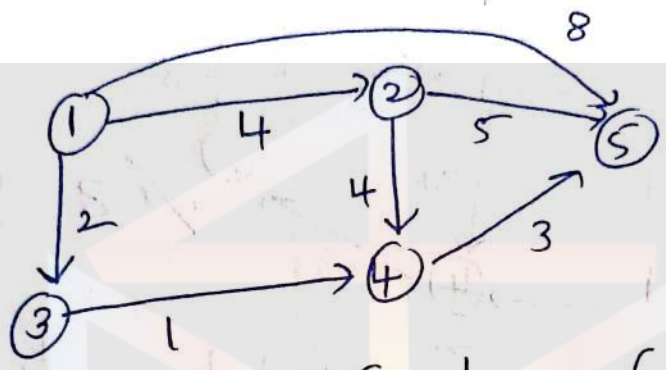
→ Shortest path algo:

→ Used to find shortest path b/n any 2 vertex

i) Dijkstra's shortest path probl

→ it is used to find shortest path b/n 2 node or vertex.

application: mostly used in network routing protocols (while sending packets).



(only for +ve weight)

= i) remove self loops & parallel edges none their.

→  → simple metho

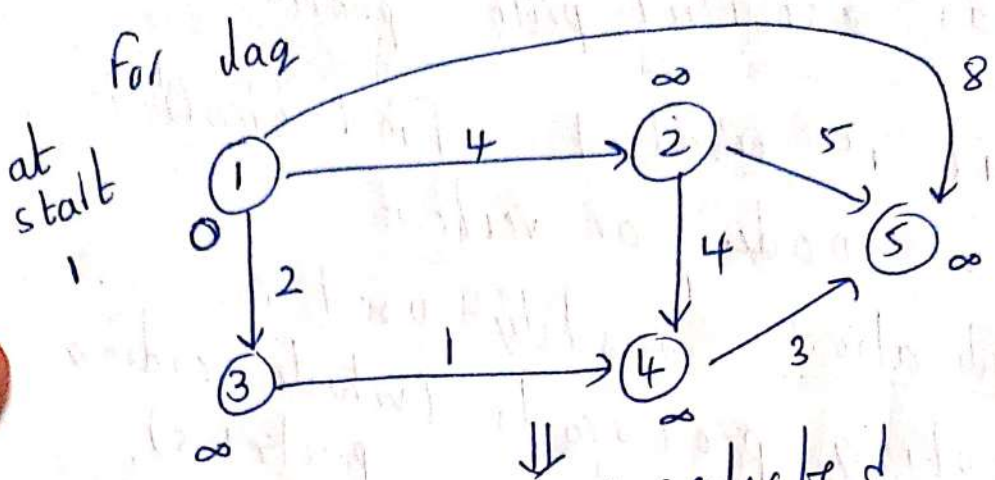
assuming 1 as source

(1)

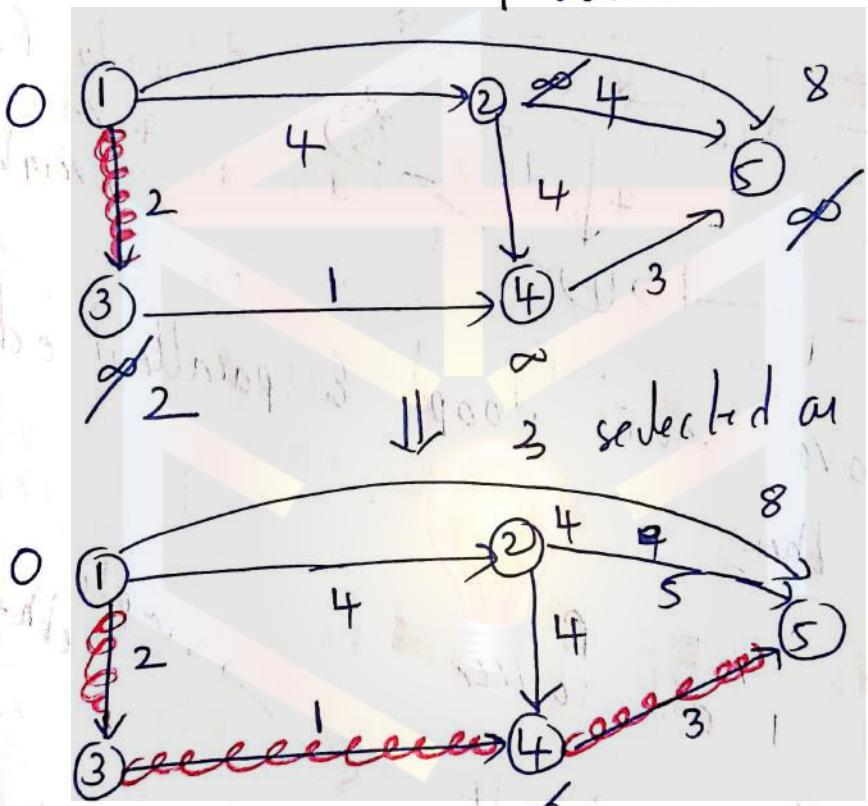
source	2	3	4	5	reached
	∞	∞	∞	∞	
1	4	2	∞	8	1 → 3 → 4 ↓ 5
3	4	2	1	8	
4	4	2	1	3	
5	4	2	1	3	

= 6 //.

For dag



⇓ 1 selected



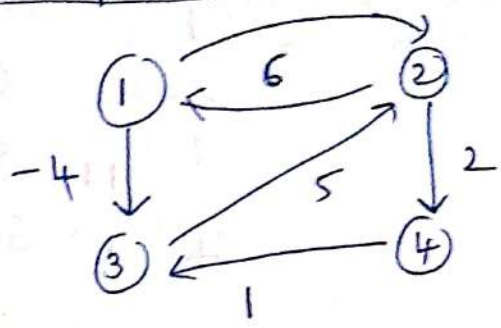
⇓ 3 selected as it is min

reached 5

1 → 3 → 4 → 5 ⇒ 6

iii) Floyd Warshall shortest path

→ -ve path allowed



step 1: Draw distance matrix.

$$D^0 \Rightarrow \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 9 & -4 & \infty \\ 6 & 0 & \infty & 2 \\ \infty & 5 & 0 & \infty \\ \infty & \infty & 1 & 0 \end{bmatrix} \end{array}$$

is path exist it's weight

edge if $(i=j)$ then 0

edge ∞

$D^0(2,3) = \infty$

$D^0(2,1) + D^0(1,3) = 2$

$6 + (-4)$

similarly continue

$$D^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 9 & -4 & \infty \\ 6 & 0 & 2 & 2 \\ \infty & 5 & 0 & \infty \\ \infty & \infty & 1 & 0 \end{bmatrix} \end{array}$$

$$D^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 9 & -4 & 11 \\ 6 & 0 & 2 & 2 \\ 11 & 5 & 0 & 7 \\ \infty & \infty & 1 & 0 \end{bmatrix} \end{array}$$

same procell

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & -4 & 3 \\ 6 & 0 & 2 & 2 \\ 11 & 5 & 0 & 7 \\ 12 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

sam
pr

$$D^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & -4 & 3 \\ 6 & 0 & 2 & 2 \\ 11 & 5 & 0 & 7 \\ 12 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

↑ final matrix

shortest path cost bln any 2 vertex can be found

Tree concept covered in DM unit 5

* Tree: A tree is a simple graph G such that there is a unique simple non-directed path b/n each pair of vertices of G.

(or)

A connected graph without any circuit is called tree.

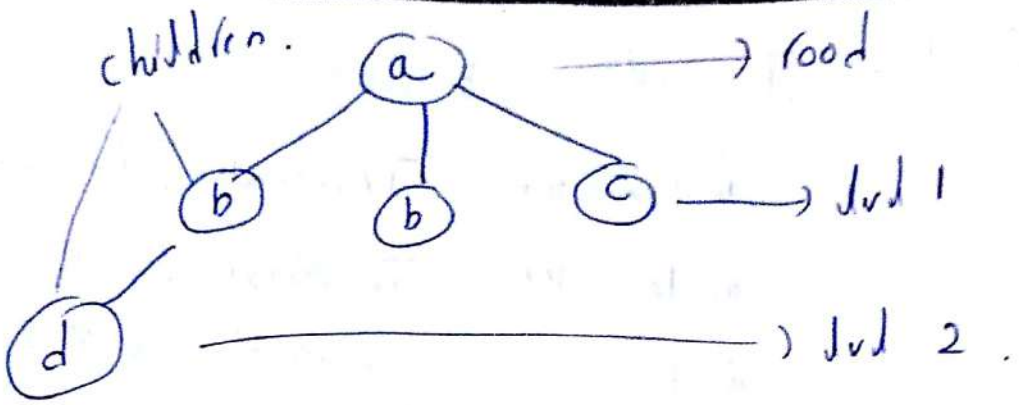
(T).

* Rooted Tree: A root tree is a tree in which there is on designated vertex called root

Directed tree: A rooted tree is a directed tree.

level: The level of a vertex v in a rooted tree is the length of the path to v from the root

Education4Fun.com

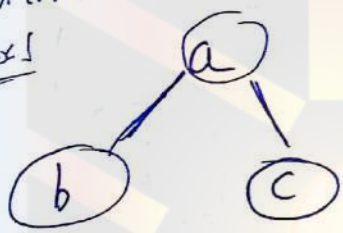


Binary tree: A tree in which there is exactly one vertex of degree two ~~& other~~

(or)

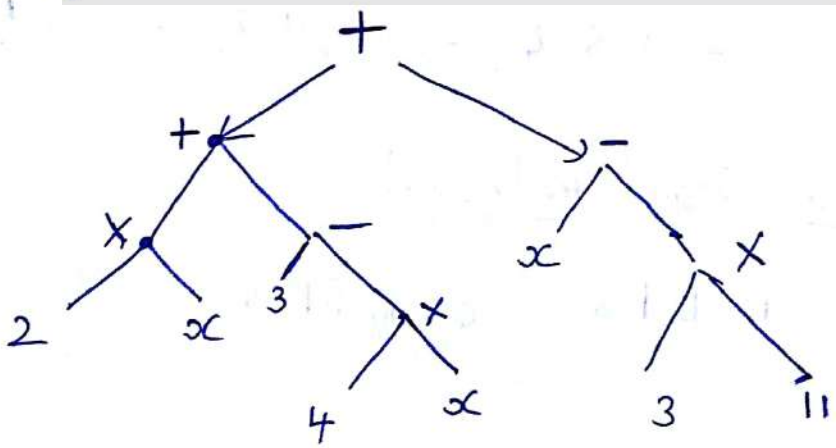
a node contain maximum 2 children.

ex:



*** From algebraic expression:**

$((2 \times x) + (3 - (4 \times x)) + (x - (3 \times 11))$



Tree Traversals: 3 types

i) Preorder Traversal (P, L, R)

ii) Inorder Traversal (L, P, R)

iii) Postorder Traversal (L, R P)

ex:

Preorder: 12, 4, 5, 1, 6, 18, 14, 99, 19, 2

Inorder: 4, 1, 6, 5, 12, 99, 2, 19, 14, 18

Postorder: 6, 1, 5, 4, 2, 19, 99, 14, 18, 12

Tree Search methods:

i) BFS & ii) DFS

i) BFS (Breadth First Search):

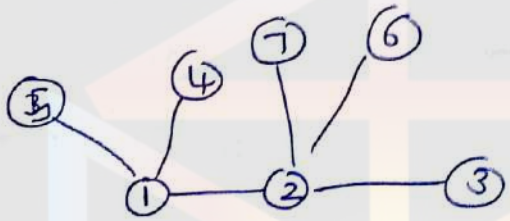
Let $G = (V, E)$ be a connected graph of order n, with vertices $v_1, v_2 \ldots v_n$ in some specified order. (Queue)

→ You can select any vertex as starting (x)

→ visit all x adjacent vertexes.

& continue step 1 & 2.

ex:



BFS order

=) 1, 2, 4, 5, 3, 6, 7 ✓

ii) DFS (Depth First Search).

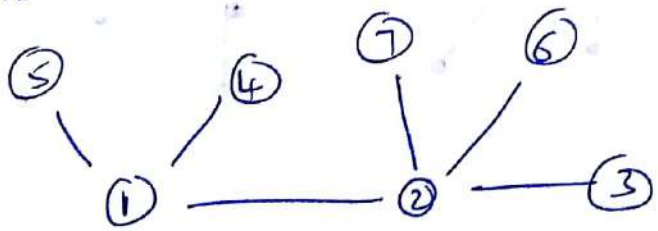
→ You can select any vertex as starting x.

→ visit one x adjacent vertex (y)

→ visit one y adjacent vertex if not found come back to x adjacent.

& continue

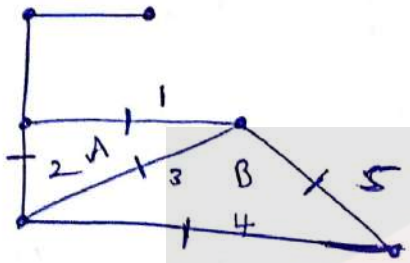
ex



1, 2, 3, 6, 7, 4, 5.

* Spanning tree: A spanning tree is a subset of Graph G which has all the vertices covered with minimum no. of edges.

ex For n vertices
⇓
(n-1) edges

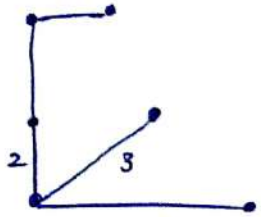


⇒ 6 vertices
&
7 edges
but,
⇒ 6 vertices ⇒ 5 edges

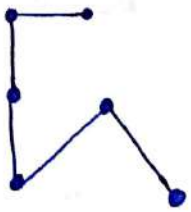
(3 × 3)
⇒ ~~3 + 3 + 3~~ − 1 ⇒ 8 possibility

From circuit A we can remove 1 edge
& From circuit B we can remove 1 edge
⇒ (3 × 3) - 9
∴ 1 common edge so −1
9 − 1 = 8

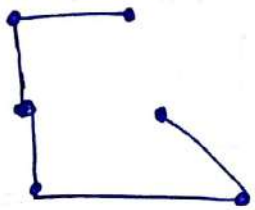
(1, 5)



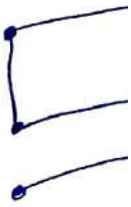
(1, 4)



(1, 3)



(2, 3



(3, 4

Mi

i)

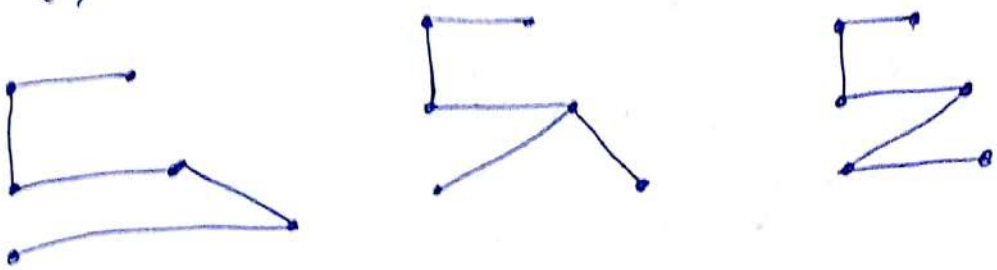
we

of

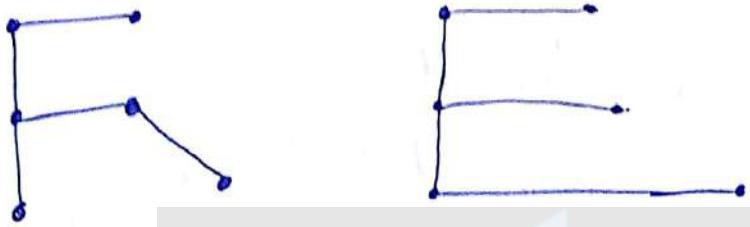
S

(

(2, 3) (2, 4) (2, 5)



(3, 4) (3, 5)



all 8 combination spanning trees ↑

Minimum cost Spanning Trees:

i) Prim's Algorithm

ii) Kruskal's Algorithm.

i) Prim's Algorithm:

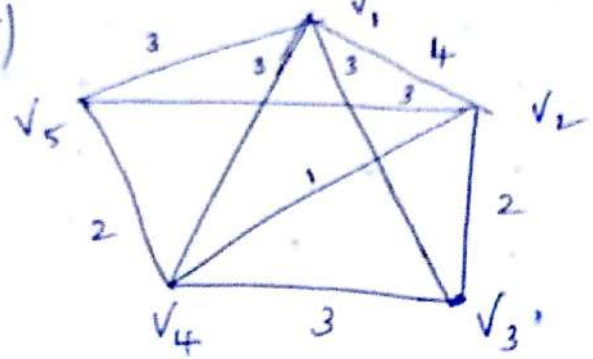
Let $G = (V, E)$ is a connected, weighted undirected graph.

Step 1: Choose any vertex V_1 of G,

Step 2: Choose an edge $e_1 = V_1 V_2$ of G such that $V_2 = V_1$ & e_1 has smallest weigh among the edges of G incident with V_1

Step 3: Continue step 2 for V_2 upto $n-1$ edges reached

ex: i)



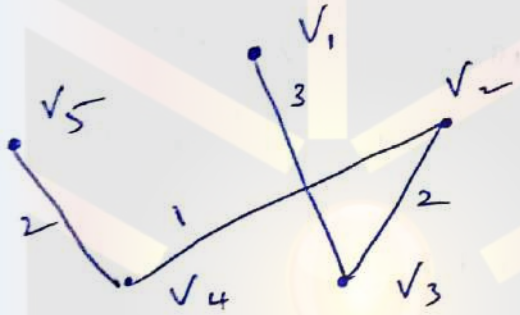
Choose V_1 search for least value

$V_1 \Rightarrow V_3$ again least to V_3

$V_1 \Rightarrow V_3 \Rightarrow V_2$ again least to V_2

$V_1 \Rightarrow V_3 \Rightarrow V_2 \Rightarrow V_4$ again least to V_4

$V_1 \Rightarrow V_3 \Rightarrow V_2 \Rightarrow V_4 \Rightarrow V_5$ //.



cost = 3 + 2 + 1 + 2 = 8 //.

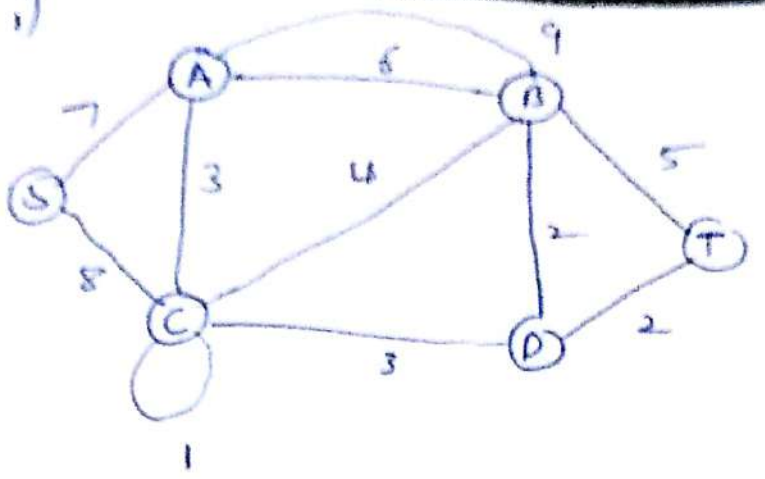
ii) Kruskal's Algorithm:

Step 1: remove all self loops & parallel edges

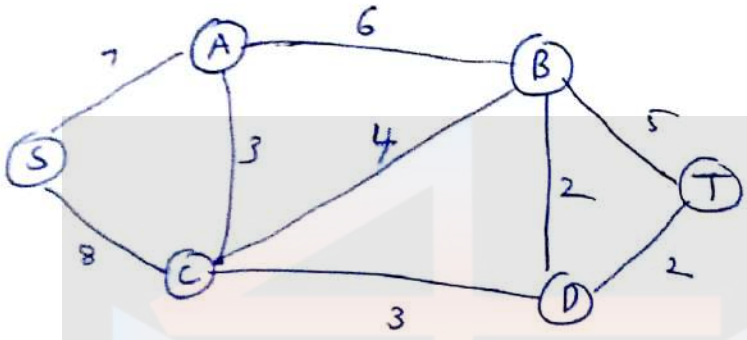
Step 2: arrange all edges in their increasing order of cost

Step 3: Add the edge which has least cost edge (If circuit are formed avoid it)

Q4 1)



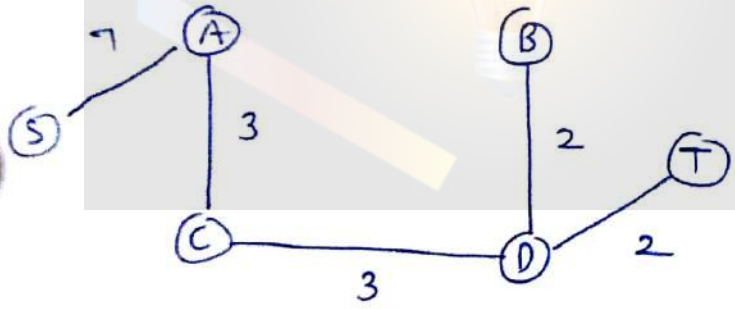
=)



=) BD DT AC CD CB BT AB SA SC

2 2 3 3 4 5 6 7 8

II)



=) 7 + 3 + 3 + 2 + 2

=) 17 min cost.