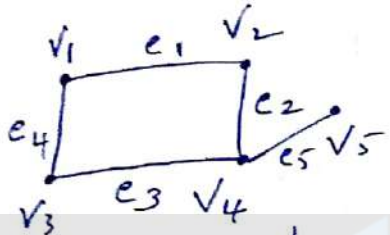
Unit-5 Graph Theory & Trees :-

* **Graph:** A graph G is a pair of sets

(V, E)

↙ vertices ↘ edges.

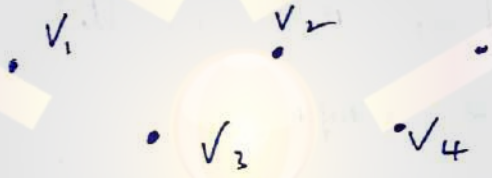
ex:



↳ 5 edges $[e_1, e_2, e_3, e_4, e_5]$

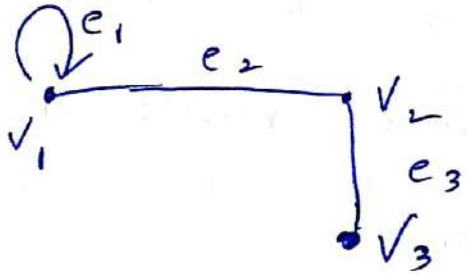
↳ 5 vertices $[v_1, v_2, v_3, v_4, v_5]$.

* **NULL Graph:** A graph in which number of edges



The NULL graph has 0 edges & 5 vertices

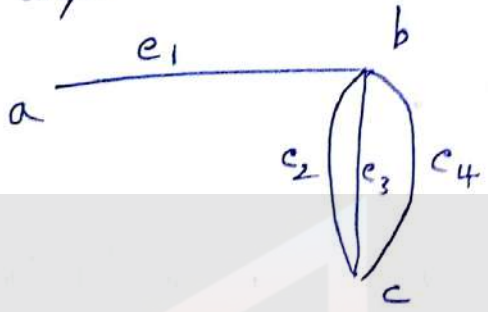
* **Self loop:** An edge joining a vertex to itself is called self loop.



e_1 is self loop.

* Parallel (or) Multiple edges

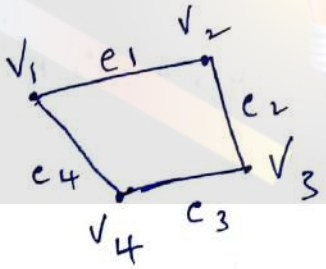
In a graph it may be possible to have more than one edge with a single pair of vertices such edges are called parallel edges



e_2, e_3, e_4 are parallel edges.

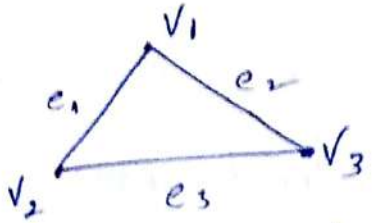
* Simple Graph:

A which contains neither self loop 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.

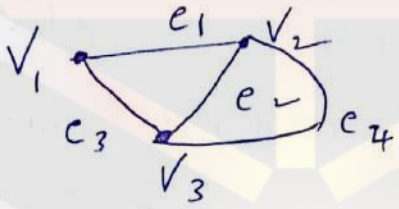


* Total no. of edges for a complete graph with 50 vertices

⇒ $^{50}C_2$ ⇒ 1225

For n vertices $^nC_2$ //.

* Multigraph: A graph containing parallel edge is called Multigraph



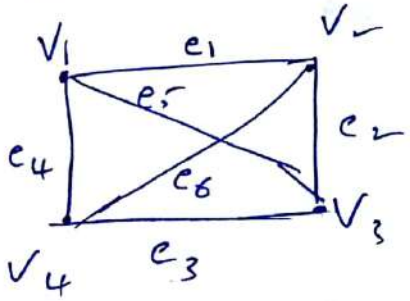
e_4, e_2 are parallel edges

* Order & Size of Graph:

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

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

ex:

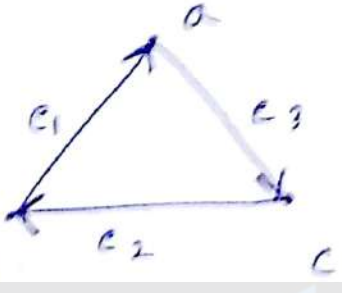


order = 4 //

size = 6 //

Directed Graph:

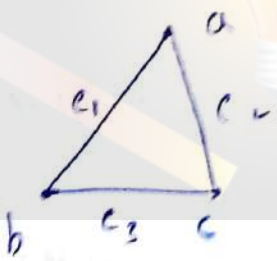
The graph in which the elements of the edge set are ordered pairs of vertices is called directed graph. or digraph



$e_1 = (b, b)$, $e_2 = (E, b)$, $e_3 = (a, c)$

Non Directed Graph:

A graph in which the elements of the edge set are unordered pairs of of vertices is called a non-Directed graph.



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

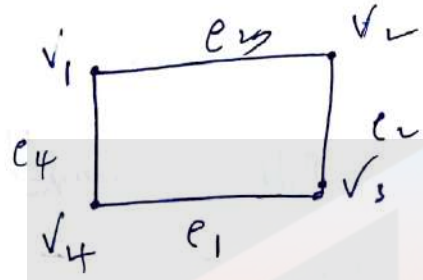
$e_2 = \{a, c\}, \{c, a\}$

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

* Adjacent vertices & Adjacent edges

→ two vertices u & v are said to be adjacent if there exists an edge. (u v).

→ 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

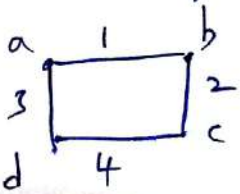
e_1 e_2, e_2 e_3, e_3 e_4, e_4 e_1 are adjacent edges

* Finite & Infinite Graph :-

A graph is a finite if both its vertex 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.



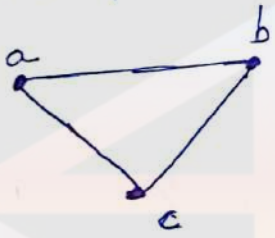
1, 2, 3, 4 are weights of the graph.

* Path :

In a path, vertices and edges may be repeated any number

The number of edges in a path is called length of the path.

A path of length zero is called trivial path.



Path	length
a - b - c	2
a - b	1
b - c	1
a	0

→ **Open path :** A path in which initial & terminal vertices are distinct is called open path.

a - b

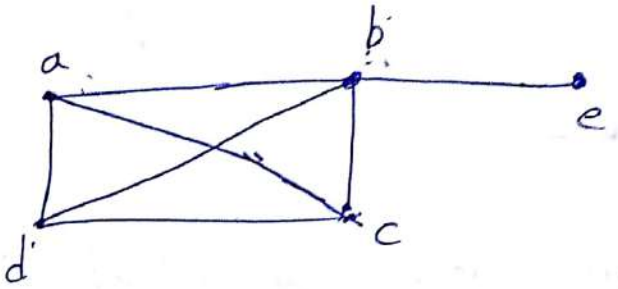
→ **Closed Path :** A path which initial & terminal vertices are same

a - b - c - a

✱ Simple Path :

A path is said to be simple if all the edges and vertices on path are different except possibly at the end points.

ex



i) a - b - c - a
ii) a - b - c - d - c - a
iii) a - b - c - d - a
iv) a - b - c - a - b

i & iii are simple path

ii & iv are not simple path.

✱ Circuit & Cycle :

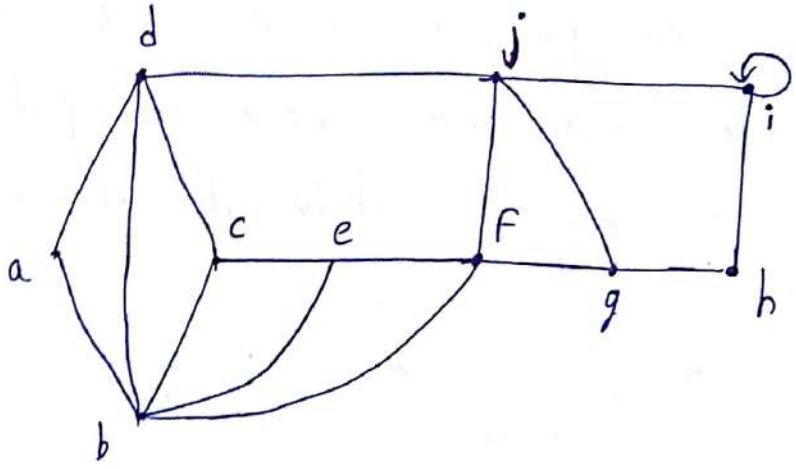
→ A path of length $d \geq 1$, with no repeated edges & whose end points are equal is called a circuit.

→ In a circuit, repetition of vertices is allowed.

→ A cycle is a circuit with no other repeated vertices except the end point.

→ every cycle is a circuit but a circuit need not be cycle.

Ex:

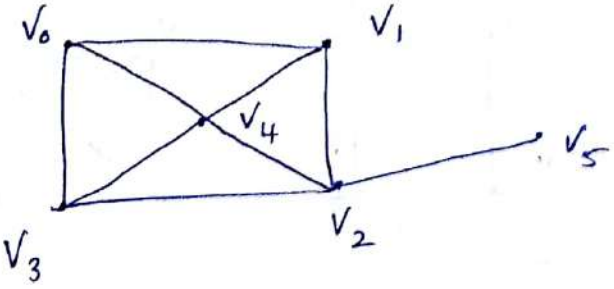


Path	length	Simple	Open	Closed	Circuit	Cycle
a-b-c-e-f-j-d-a	7	Y	N	Y	Y	Y
b-c-e-f-g-i-f-b	7	N	N	Y	Y	N
a-b-a	2	~~Y~~	N	Y	N	N
a-b-c-b-a	4	Y	N	Y	N	N
i-i	1	Y	N	Y	Y	Y
a	0	Y	N	Y	N	N
d-b-c-d	3	Y	N	Y	Y	Y
e-f-g-j-f-b	5	N	Y	N	N	N

* Edge Disjoint Path & Vertex Disjoint Paths:

Two paths in a graph are said to be edge disjoint if they have no common edges, but they have a common vertices

Two paths in a graph are said to be vertex disjoint if they have no common vertices

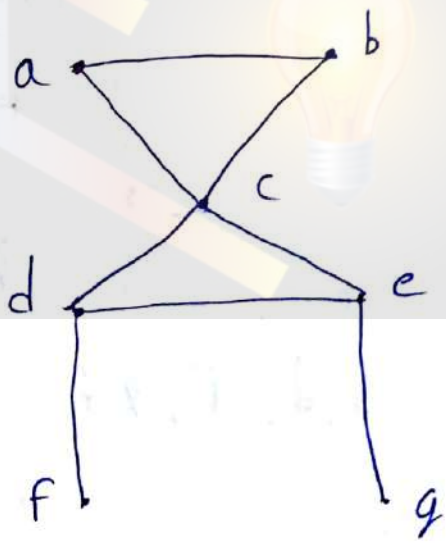


→ (V_0, V_4), (V_4, V_3), (V_1, V_4) (V_4, V_2) are edge-disjoint

→ $\{(V_3, V_2), (V_2, V_5)\}$ & $\{(V_0, V_4)(V_4, V_1)\}$ are vertex disjoint.

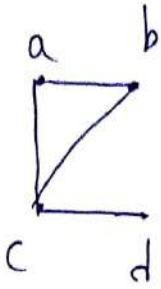
✱ Connected Graph :-

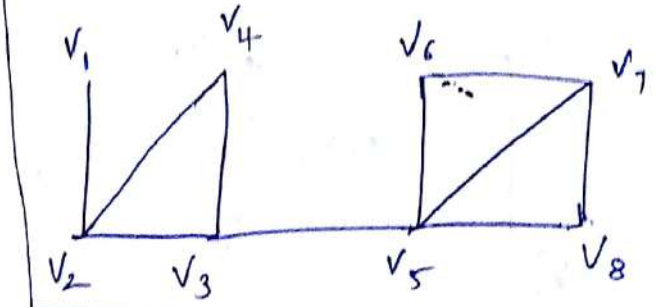
A undirected graph is connected if there is a path b/n every pair of distinct vertices of the graph



✱ **Cut Vertex** :- It is a vertex by which if we remove that vertex then the graph will be disconnected graph.

ex:





~~by~~

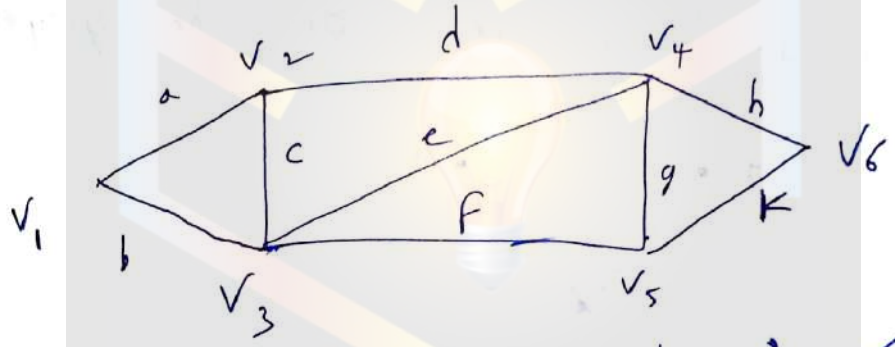
Here C is cut vertex

V_2, V_3, V_5 are cut vertex.

* Cut set:

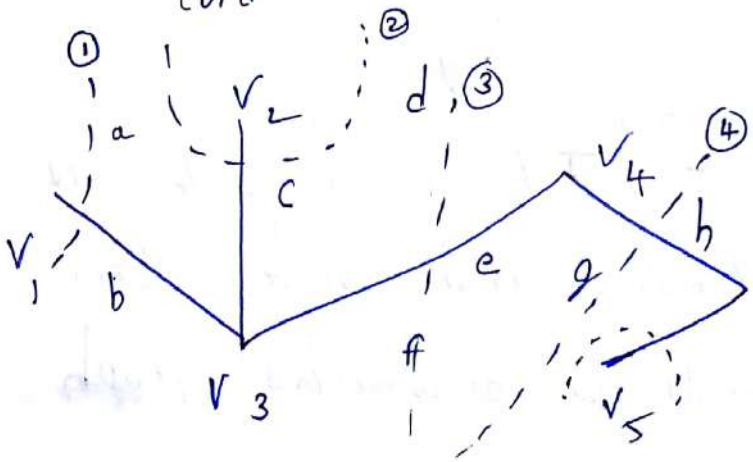
→ Cut set partition all the vertices into two disjoined sets

→ Cut set always contain only one branch & rest of edges are chords.



branches = { b, c, e, h, k } ✓

cord = { a, d, F, g } ✓



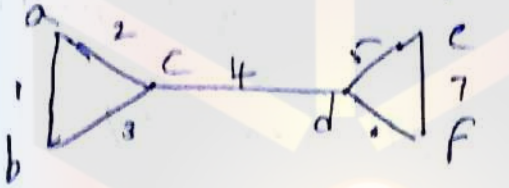
① {a, b}
② {a, c, d}
③ {d, e, F}
④ {h, g, F}
⑤ {F, g, k}

are cut set.

* Edge Connectivity:

each cut set of a connected graph G consists of a certain number of edges. The number of edges in the smallest cut set is defined as the edge connectivity.

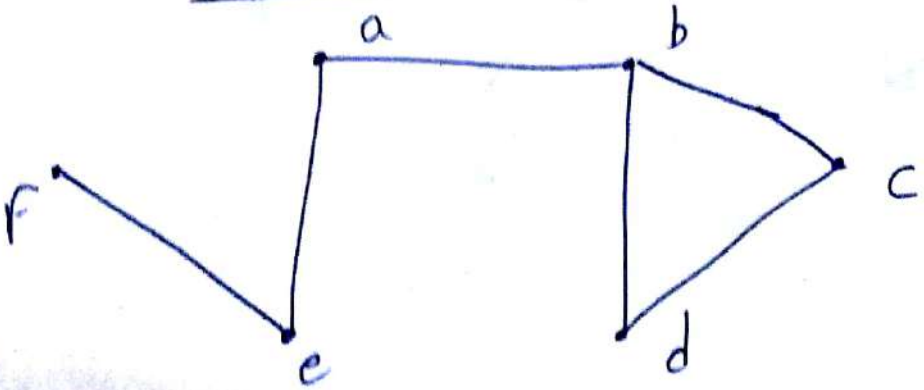
* Bridge: It is an edge which connects the two vertices & removal of such edge disconnects the graph into two disjoint graphs.



4 is Bridge edge.

* Vertex Connectivity: The vertex connectivity of a connected graph G is the smallest number of vertices whose removal disconnects G.

ex i) Find cut vertices

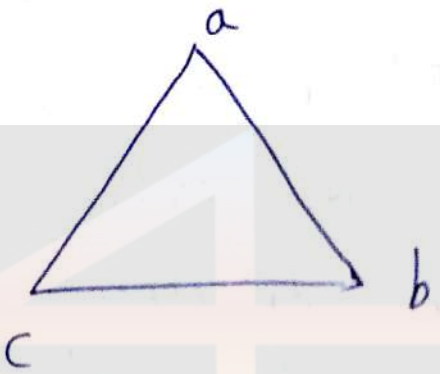


b & c are cut vertices.

✱ Cycle Graph :

A cycle graph of order n is a connected graph whose edges form a cycle of length n & is denoted by C_n

ex



✱ Wheel Graph : A wheel graph of order n is a graph obtained by joining a single new vertex to each vertex of cycle graph (C_{n-1}) of order $(n-1)$. denoted by W_n

(like a wheel)

W_4 W_3

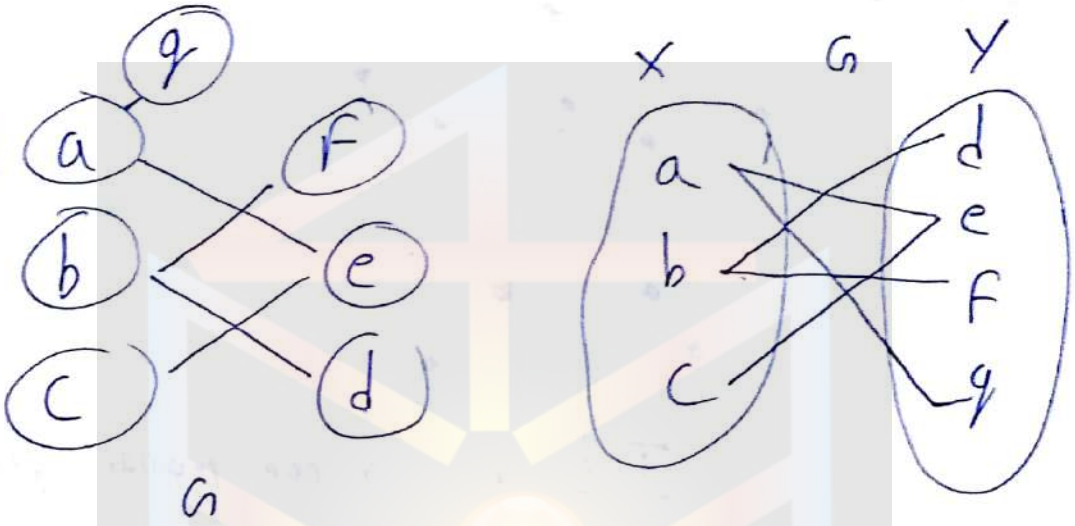
Bipartite Graph :-

→ It is a simple graph

→ In which the set of vertices can be partitioned into two sets X and Y

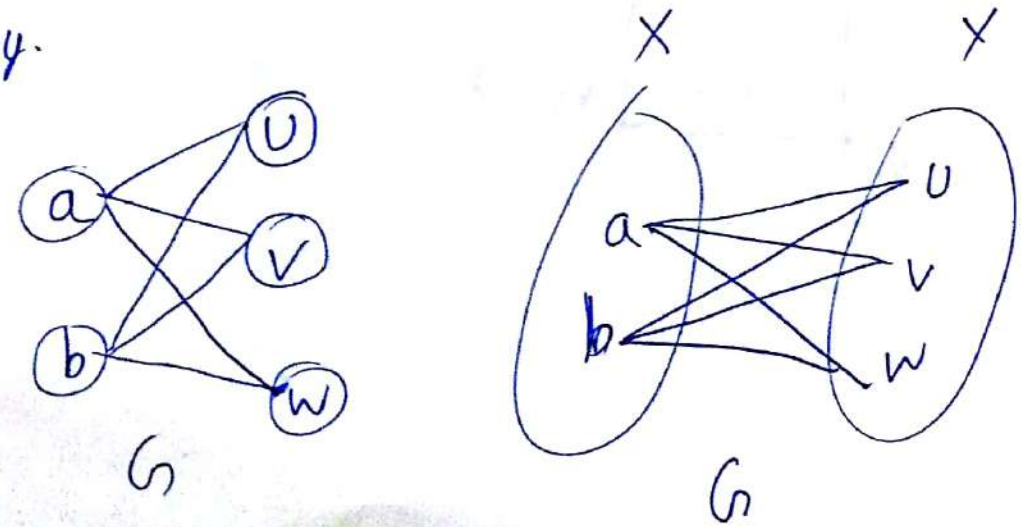
∴ G (X, Y, E)

ex:



Complete Bipartite Graph :-

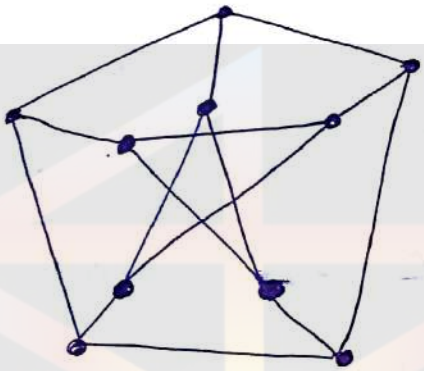
It is a Bipartite Graph in which ~~each & any~~ their is an b/n every vertex in X & every vertex in Y.



* K - Partite Graph:

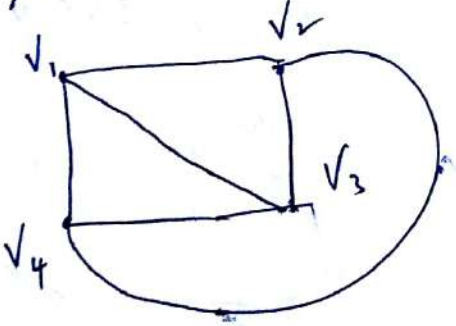
It is a k partite graph we will be having dividing into that many disjoint subjects.

* Peterson Graph: It is undirected graph with 10 vertices & 15 edges



It is a non planar graph.

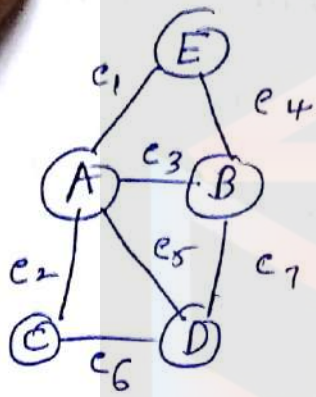
* Planar graph is a graph that can be drawn in the plane without any edge crossing.



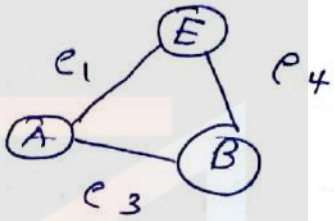
★ **Subgraph**: Suppose there are two graph $H = (V_2, E_2)$

$G = (V_1, E_1)$

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



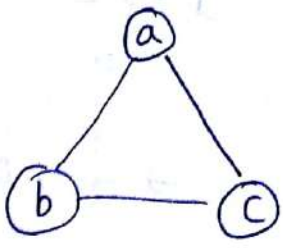
(G)



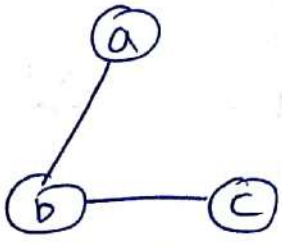
(H)

★ **Component** Let G be a graph & let v be a vertex in G. The subgraph G' of G consisting of all edges & vertices in G is called a Component graph

eg



(G)



(G')

✱ Euler Graph:

max two odd degree

Euler path :- It is a path that traverse each edge exactly once & only path.

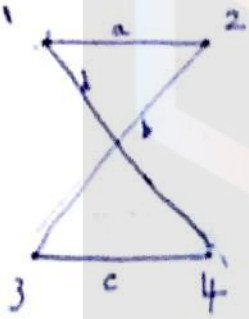
Euler graph:- a graph that contains an euler path is called euler

even degree

Euler circuit:- First & last vertex are same It is a circuit that traverse each edge exactly one & only one

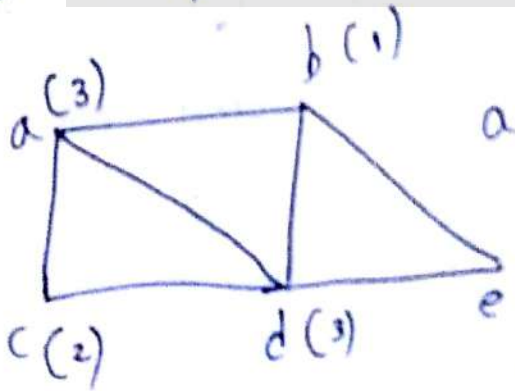
→ Vertex can be repeated but not edge.

ex



a → b → c → d

ex:



a → b → d → b → a

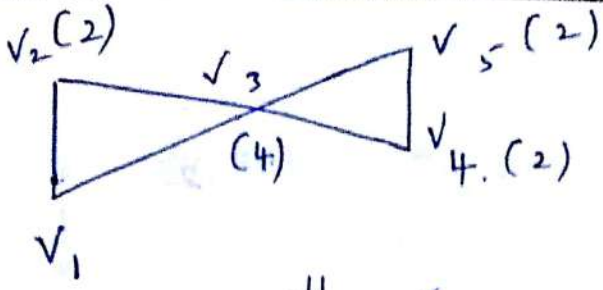
d → e

or

a → c → d → b → e → d → a

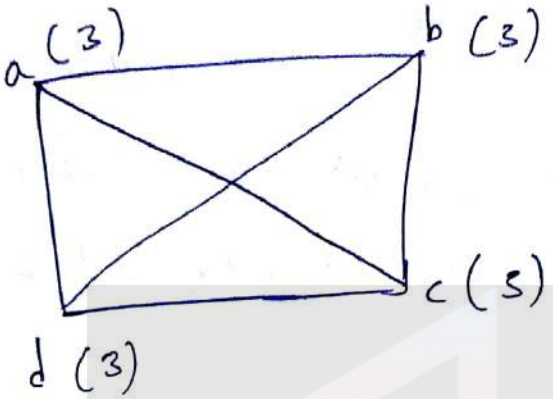
a, d → two odd

(euler path).



all even
euler circuit.

$v_1 \to v_2 \to v_3 \to v_4$
$\downarrow$
v_5
$\downarrow$
v_3
$\downarrow$
v_1

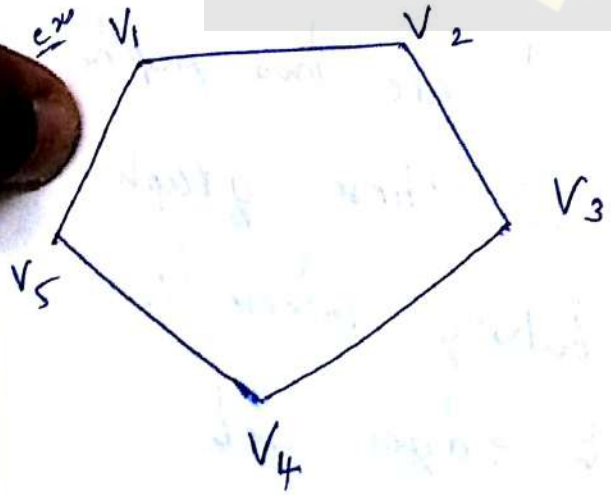


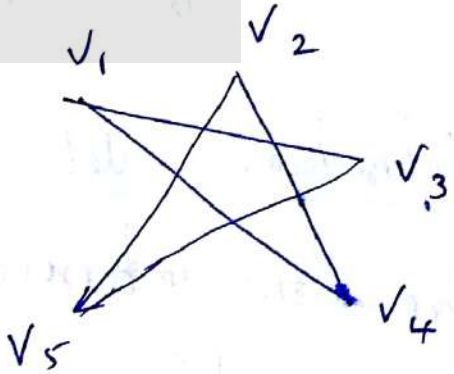
all odd
not a euler graph.

✱ Complement of a Graph:

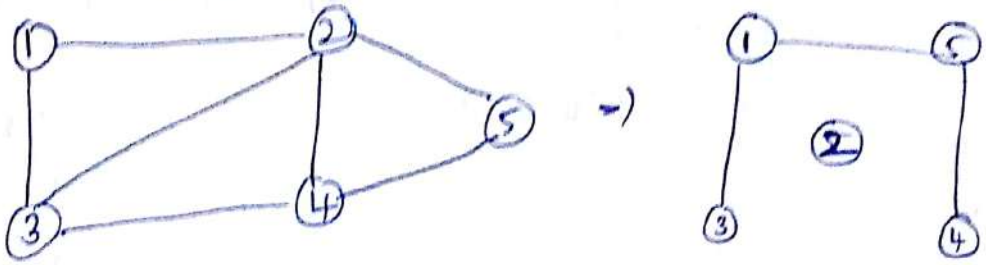
→ It is a simple graph G
→ having all vertices of G
→ in which there is an edge b/n two vertices u & w if & only if there is no edge b/n u & w. in G.

ex





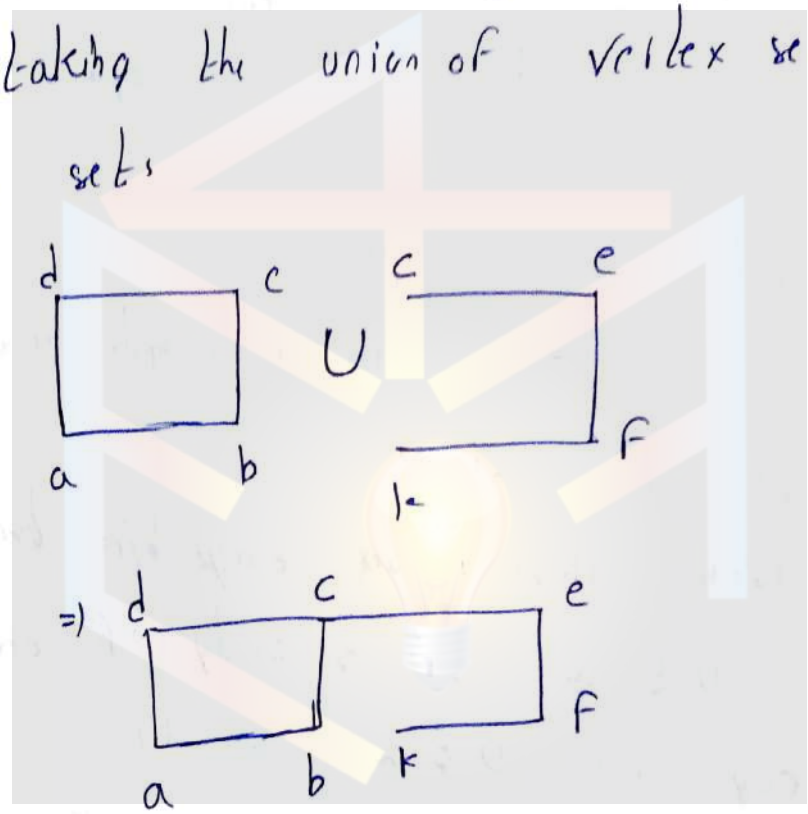
→a
↓
b



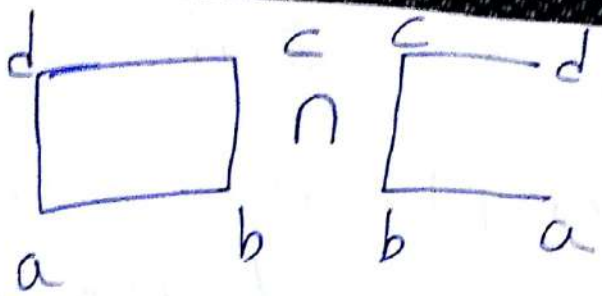
* Union & Intersection of Graph :-

Union: If G & G' are two graphs then the union of these graph are obtained by taking the union of vertex set & edges sets

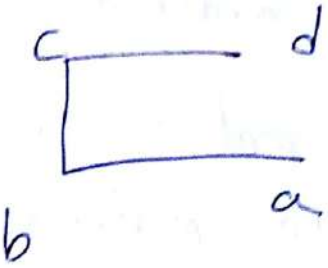
Ex.



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



=)



* Ring sum of graphs :- $G_1 \Delta G_1$

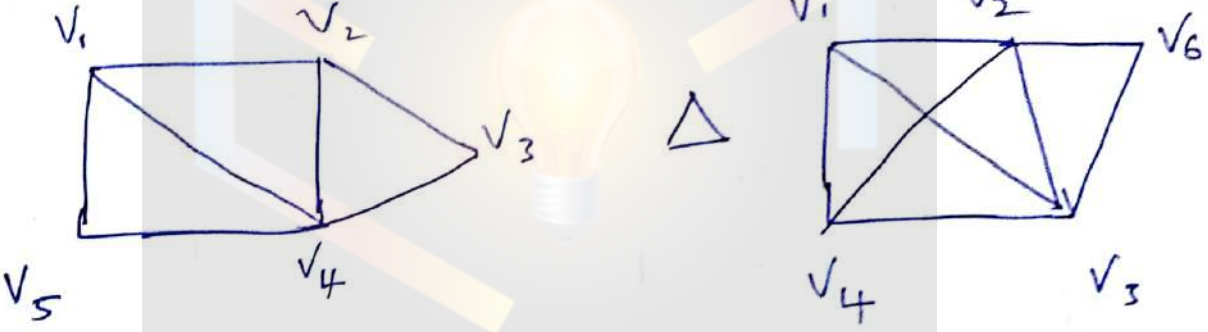
=)

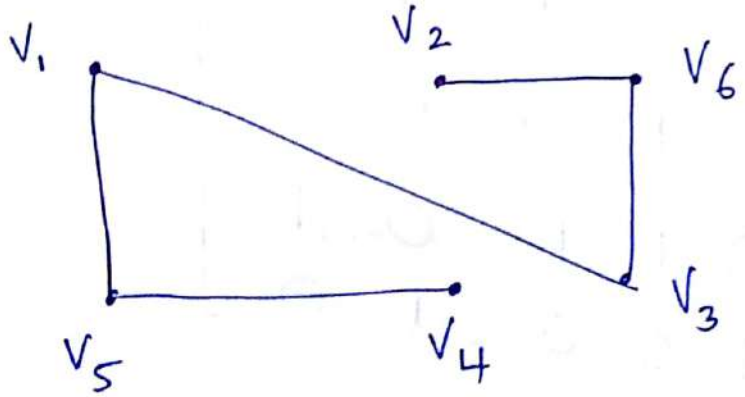
$$G_1 \Delta G_2 = (V_1 \cap V_2, E_1 \Delta E_2)$$

$V_1 \cap V_2$ → vertices

$E_1 \Delta E_2$ → edges.

ex





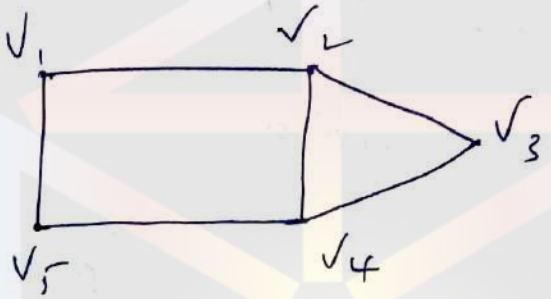
Education4Fun.com

✱ Adjacency Matrix of a Graph :

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

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

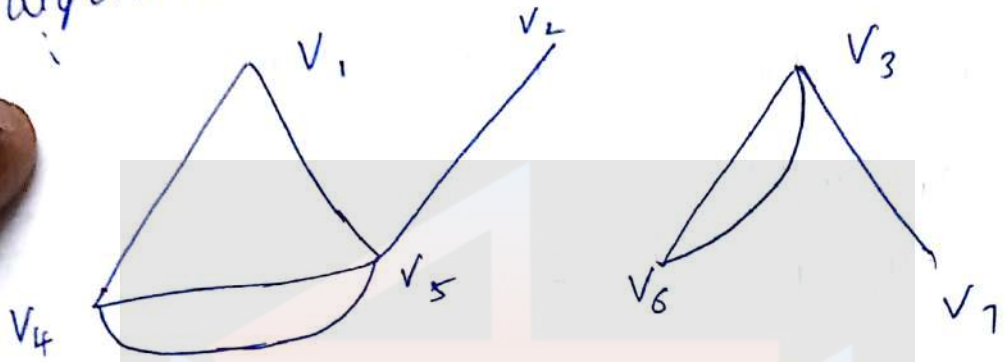
ex:



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

Fusion of Graphs :-

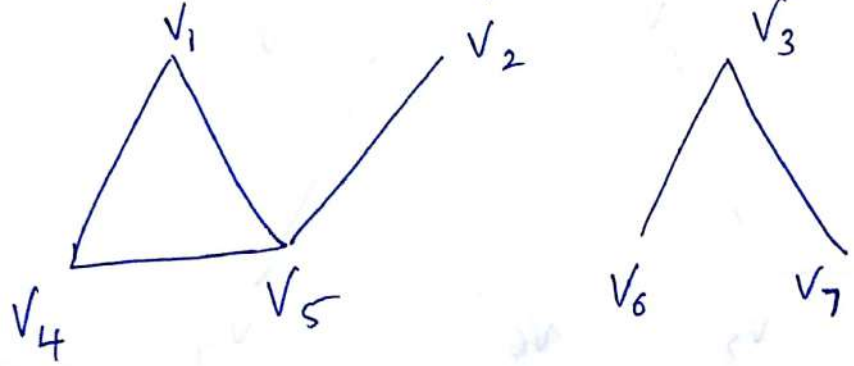
ex:- i) Given below is the adjacency matrix of graph G with 7 vertices labled $v_1, v_2, v_3, v_4, v_5, v_6, v_7$ use Fusion algorithm to check the connectedness.



A_G =

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	0	0	1	1	0	0
v_2	0	0	0	0	1	0	0
v_3	0	0	0	0	0	2	1
v_4	1	0	0	0	2	0	0
v_5	1	1	0	2	0	0	0
v_6	0	0	2	0	0	0	0
v_7	0	0	1	0	0	0	0

Step 1:- Remove all parallel edges & self loops



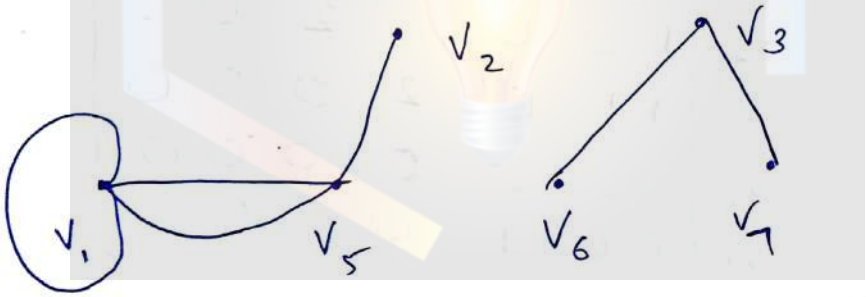
A_G =

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	0	0	1	1	0	0
v_2	0	0	0	0	1	0	0
v_3	0	0	0	0	0	1	1
v_4	1	0	0	0	1	0	0
v_5	1	1	0	1	0	0	0
v_6	0	0	1	0	0	0	0
v_7	0	0	1	0	0	0	0

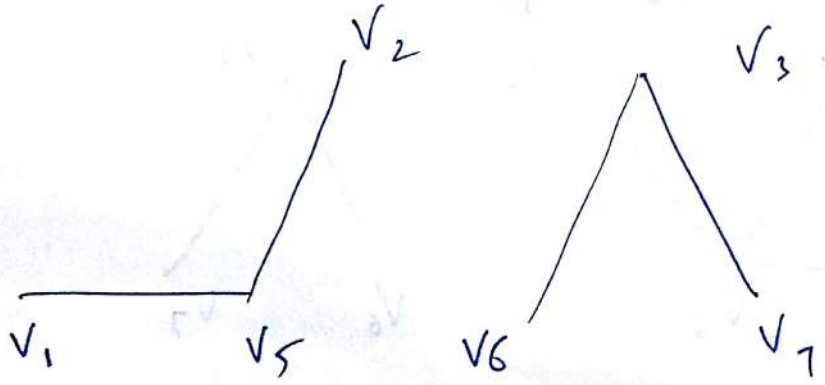
Step 2:- Fusing v_1 with v_4

$v_1 = v_1 + v_4$

	v_1	v_2	v_3	v_5	v_6	v_7
v_1	1	0	0	2	0	0
v_2	0	0	0	1	0	0
v_3	0	0	0	0	1	1
v_5	2	1	0	0	0	0
v_6	0	0	1	0	0	0
v_7	0	0	1	0	0	0



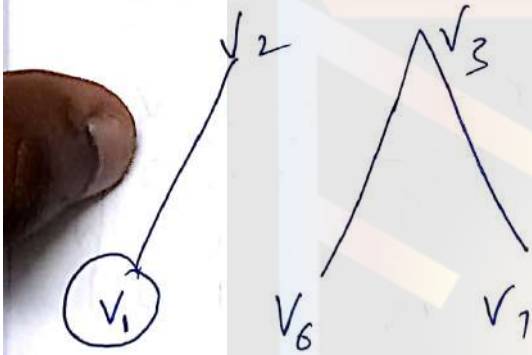
Step 3:- Remove parallel edges & self loops



	V_1	V_2	V_3	V_5	V_6	V_7
V_1	0	0	0	1	0	0
V_2	0	0	0	1	0	0
V_3	0	0	0	0	1	1
V_5	1	1	0	0	0	0
V_6	0	0	1	0	0	0
V_7	0	0	1	0	0	0

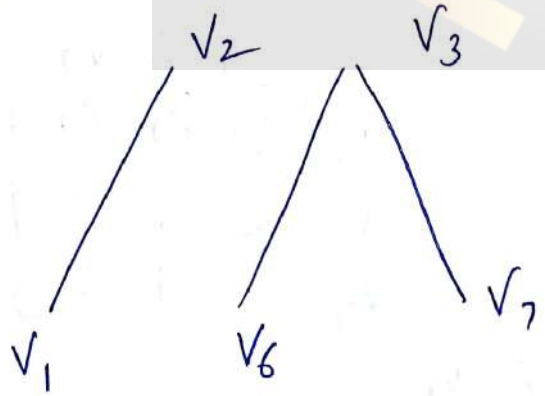
Step iv Fusion with V_1 with V_5

$V_1 = V_1 + V_5$



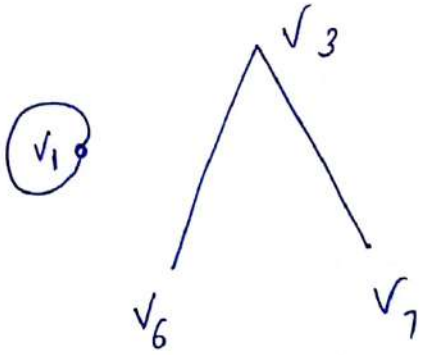
	V_1	V_2	V_3	V_6	V_7
V_1	1	1	0	0	0
V_2	1	0	0	0	0
V_3	0	0	0	1	1
V_6	0	0	1	0	0
V_7	0	0	1	0	0

Step v. Remove parallel & self-loops



	V_1	V_2	V_3	V_6	V_7
V_1	0	1	0	0	0
V_2	1	0	0	0	0
V_3	0	0	0	1	1
V_6	0	0	ϕ	0	0
V_7	0	0	1	0	0

Step 6: Fusing V_1 & V_2

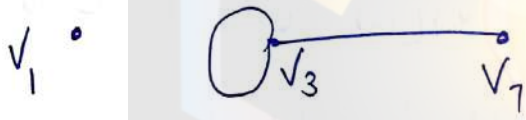


	V_1	V_3	V_6	V_7
V_1	1	0	0	0
V_3	0	0	1	1
V_6	0	1	0	0
V_7	0	1	0	0

Step 7: remove parallel edges & ~~self~~ self loops

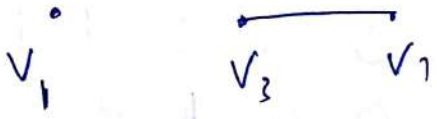
	V_1	V_3	V_6	V_7
V_1	0	0	0	0
V_3	0	0	1	1
V_6	0	1	0	0
V_7	0	1	0	0

Step 8: Fussion V_3 with V_6



	V_1	V_3	V_7
V_1	0	0	0
V_3	0	1	1
V_7	0	1	0

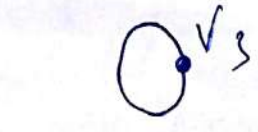
Step 9: remove parallel edges & self-loops



	V_1	V_3	V_7
V_1	0	0	0
V_3	0	1	1
V_7	0	1	0

Step 10: Fusion V_3 with V_1

$V_3 = V_3 + V_7$



	V_1	V_3
V_1	0	0
V_3	0	1

Step 11: remo. parallel edges & self loops

v_1 v_3 $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

Hence that original graph G has 2 connected components

Isomorphism :-

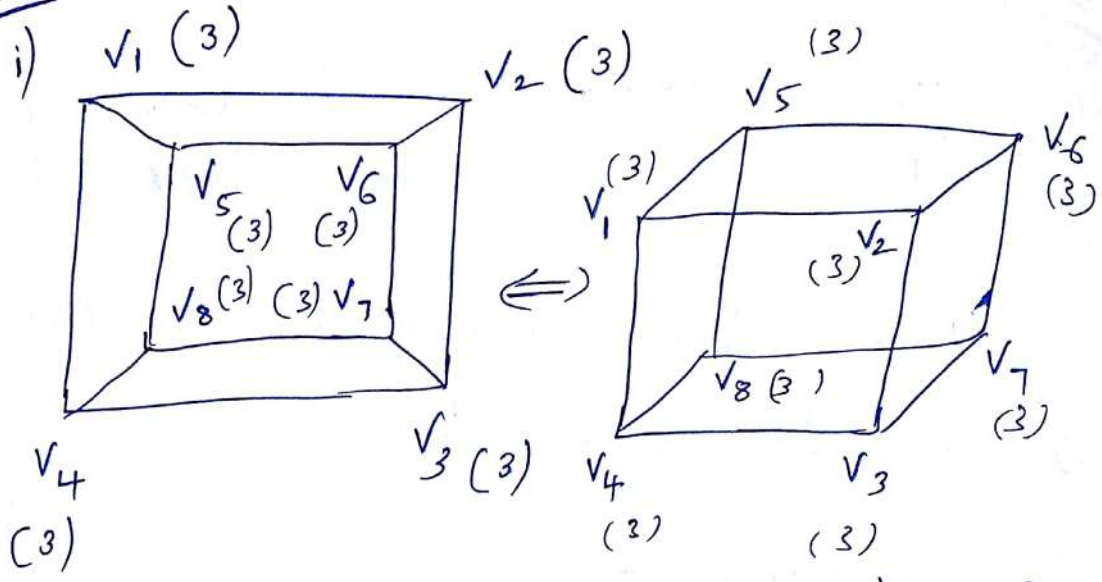
→ Two graphs are said to be isomarphic if they are perhaps the same graphs just drawn differently with different names i.e they have identical behaviour for any graph-theoretic property.

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.

ex:

i)



→ having same no. of edges, vertices & order.

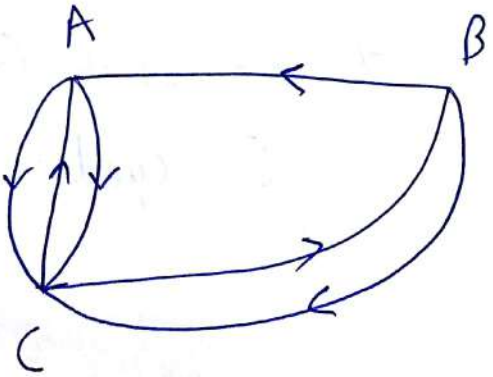
Hence it is similar // Follow Isomorphic.

* Multigraph :-

Multigraph consists of vertices & undirected edges between these vertices with multiple edges b/n pair of vertices allowed

every simple graph is also a multigraph

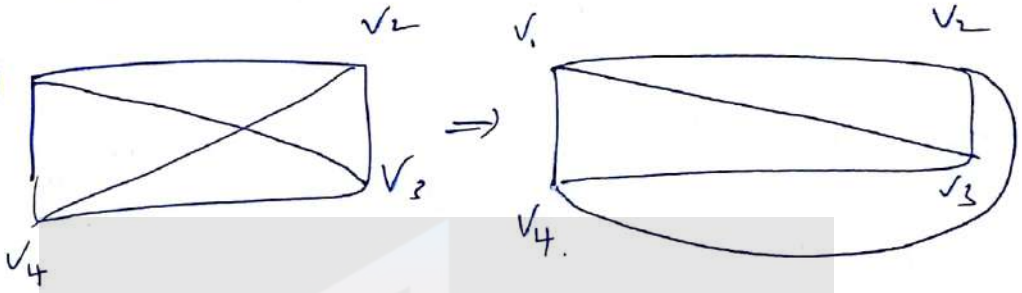
ex:



De Bruijn Sequence :-

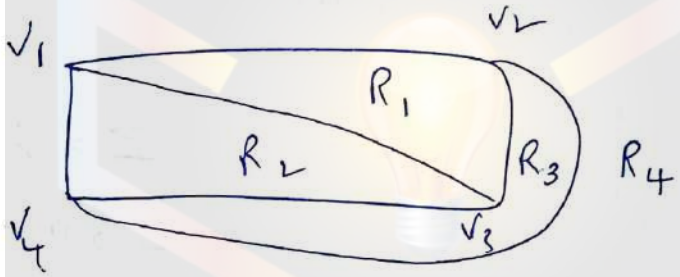
* **Planar Graph:** It is a graph that can be drawn in the plane without any edge crossing

ex:



* **Faces (or) Regions :**

A planar graph can be divided into the contiguous region called Faces



* **Euler's Formula:** Let G be a connected planar simple graph with e edges & v vertices. Let r be the no. of region

$$r = e - v + r.$$

ex

$$r = 6 - 4 + 2$$

$$= 4$$ Hence proved

* Planarity Testing algorithm:

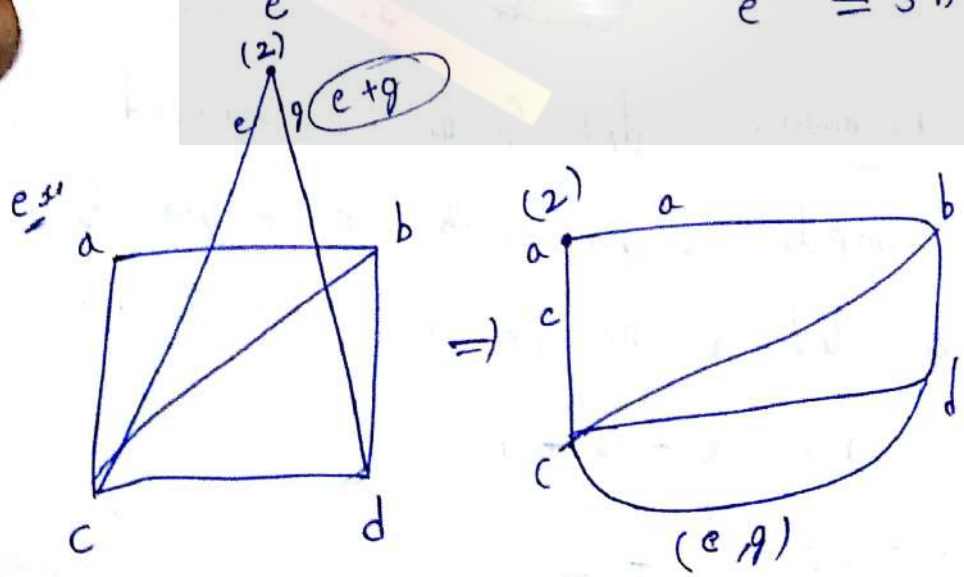
→ If graph is disconnected then it has several components, take 1 at a time

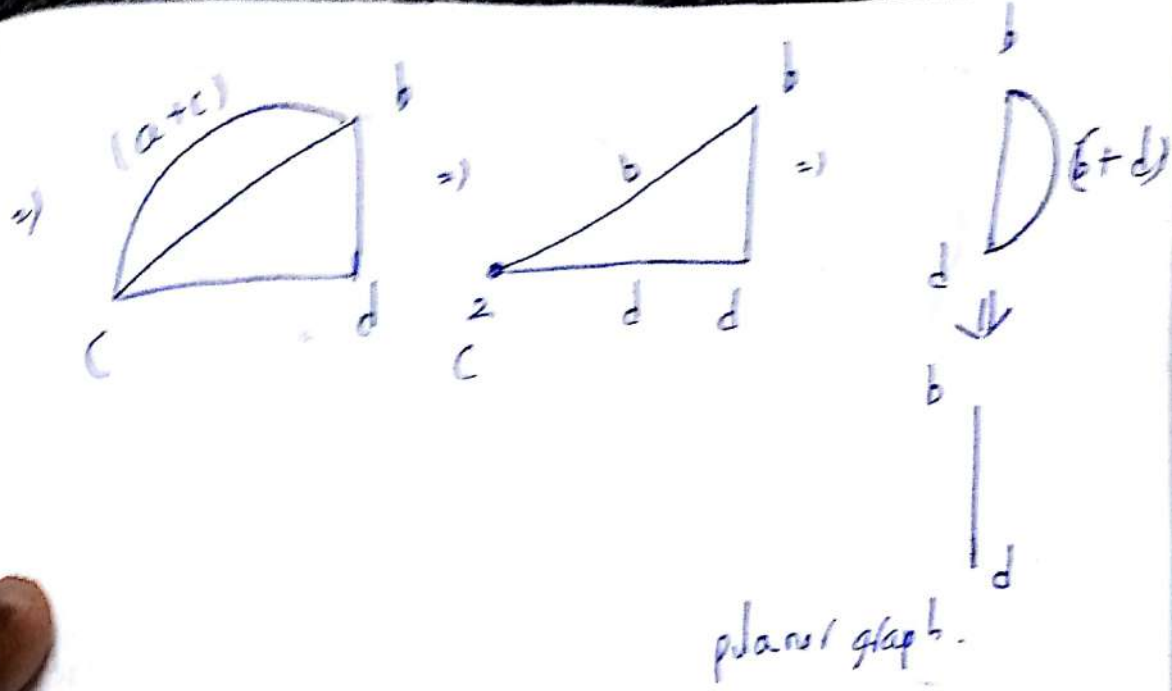
→ If graph has parallel edges or self loops, remove them.

→ If there is any vertex in the graph of degree 2, we will merge them.

After finishing check it.

A single edge
(or)
a complete graph with 4 vertices
or
A graph with $n \geq 5$; $e \geq 7$
$e \leq 3n - 6$





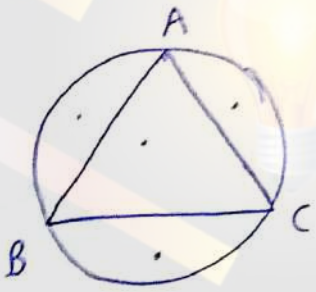
planar graph.

* Dual Graph :-

→ It has vertex for each Face

→ Has a edge whenever two Face are separated by an edge.

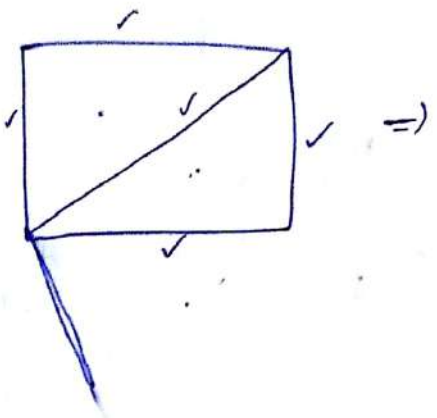
Ex:- i)



Draw Dual Graph

ii)

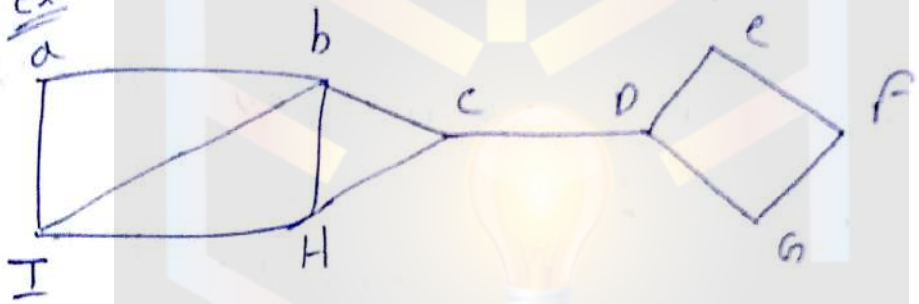
Education4Fun.com

ii)  ⇒ 

Hamiltonian :

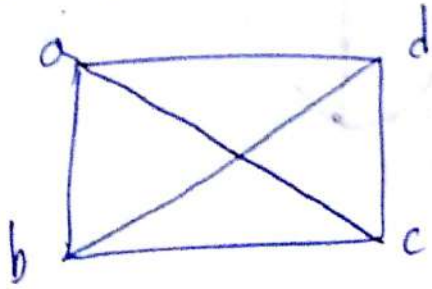
→ **Hamiltonian path** : that is path through a graph that goes through every vertex once & only once.

ex!

i) 

A → I → H → B → C → D → E → F → G (✓).

→ **Hamiltonian Circuit** : It is a hamiltonian path which starts & stops at the same vertex.

ex! 

a → b → c → d → a (✓)

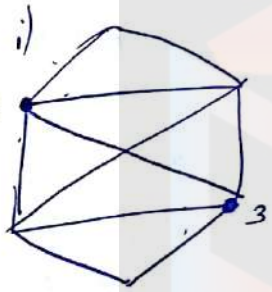
→ Hamiltonian Closure of a Graph :

The graph with vertex set $V(G)$ obtained from G by iteratively adding edges joining pairs of non adjacent vertices whose degree sum is at least n, until no such pair remains

If $d(u) + d(v) \geq n$.

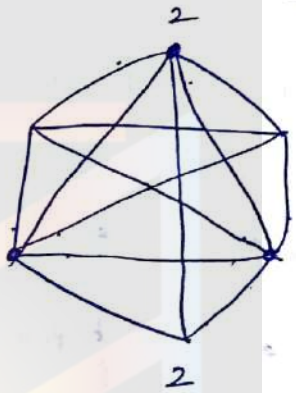
$n = 6$

ex i)



$3 + 3 = 6$ ✓

$3 + 3 = 6$ ✓ ⇒



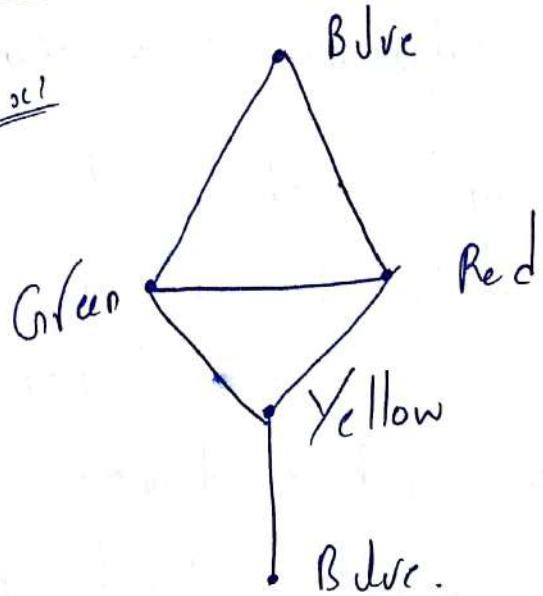
$4 + 2 = 6$
$4 + 2 = 6$
$4 + 2 = 6$ ✓

✓.

✱ Graph Coloring :

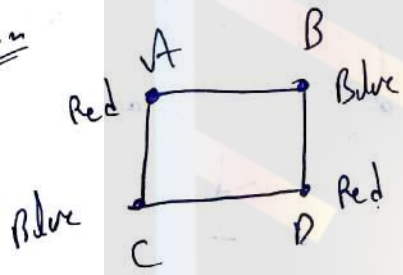
Given a planar or non planar graph G, if we assign colors to its vertices in such a way that no two adjacent vertices have the same color then graph G is properly colored.

ex:

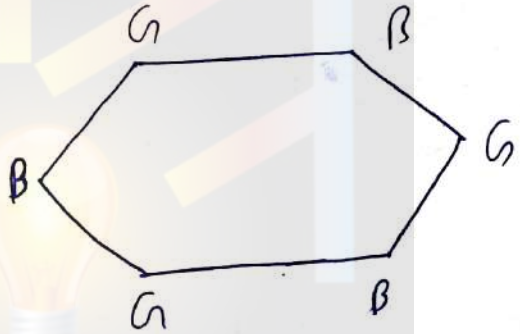


* Chromatic Numbers :- The minimum number of colors required to color a graph G is called Chromatic Number. $\chi(G)$

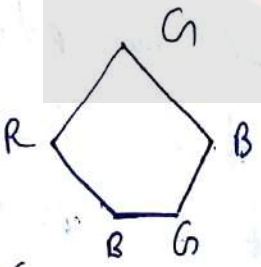
ex



$\chi(G) = 2$



$\chi(G) = 2$



$\chi(G) = 3$

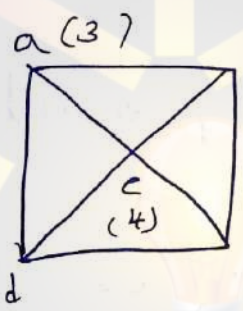
Welsh Powell Algorithm:

→ To color a graph G, first order the vertices according to decreasing degree.

Step 1: Use first color to color the first vertex & color in sequential order i.e, each vertex which is not adjacent to a previously colored vertex.

Step 2: Repeat the process using the second color ..... 3rd color ... nth color.

ex: i)



→ (4, 3, 3, 3, 3)

↓↓

(e, a, b, c, d)

color 1 to vertex e
color 2 to vertex a, c
color 3 to vertex b, d

$\chi(G) = 3$

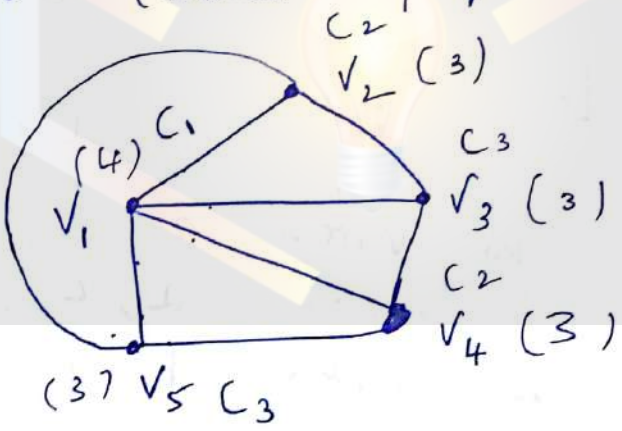
✱ Chromatic Polynomial :-

Let G be a graph & $F(G, x)$ be the no. of different colourings of a graph with x or fewer colours then $F(G, x)$ can be expressed as a polynomial in x.

∴ the colour

$$= a_1\lambda + a_2\frac{\lambda(\lambda-1)}{2!} + a_3\frac{\lambda(\lambda-1)(\lambda-2)}{3!} + a_n\frac{\lambda(\lambda-1)\ldots(\lambda-n-1)}{n!}$$

ex: i) Find the chromatic polynomial of graph



Chromatic number

$\Rightarrow (4, 3, 3, 3, 3) \Leftrightarrow (V_1, V_2, V_3, V_4, V_5)$

$\Rightarrow V_1 \Rightarrow C_1$

$\Rightarrow V_2 \Rightarrow C_2 \Leftarrow V_4$

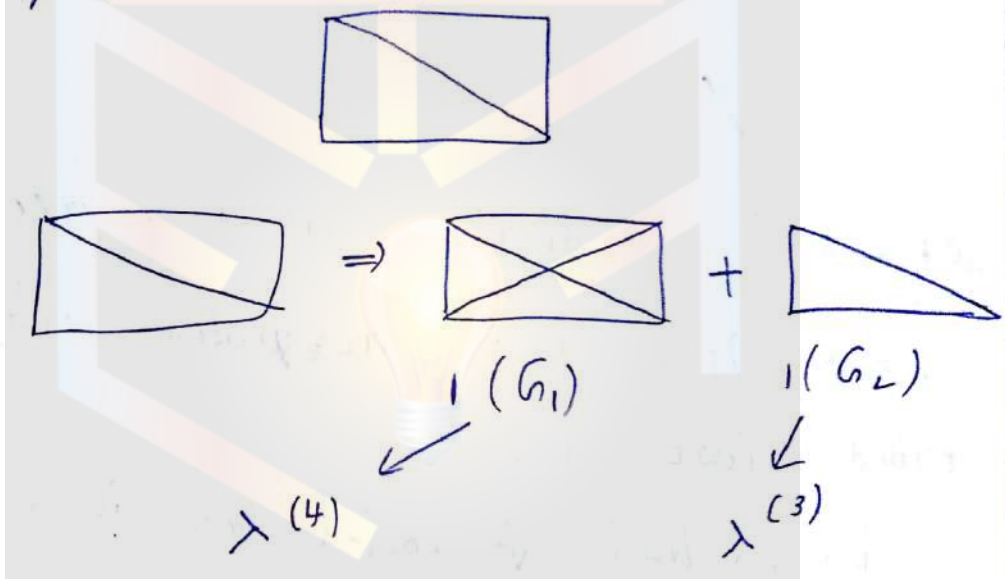
$\Rightarrow V_3 \Rightarrow C_3 \Leftarrow V_5$

$\therefore \chi(G) = 3$

Total 5 vertice

$$P_5(\lambda) = a_1\lambda + a_2\frac{\lambda(\lambda-1)}{2!} + a_3\frac{\lambda(\lambda-1)(\lambda-2)}{3!} + a_4\frac{\lambda(\lambda-1)(\lambda-2)(\lambda-3)}{4!} + a_5\frac{\lambda(\lambda-1)(\lambda-2)(\lambda-3)(\lambda-4)}{5!}$$

ii) Find the chromatic polynomial of a graph.



$$\Rightarrow \lambda(\lambda-1)(\lambda-2)(\lambda-3) + \lambda(\lambda-1)(\lambda-2)$$

$$\Rightarrow \lambda(\lambda-1)(\lambda-2)[\lambda-3+1]$$

$$\Rightarrow \lambda(\lambda-1)(\lambda-2)^2 \text{ //}$$

* 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.

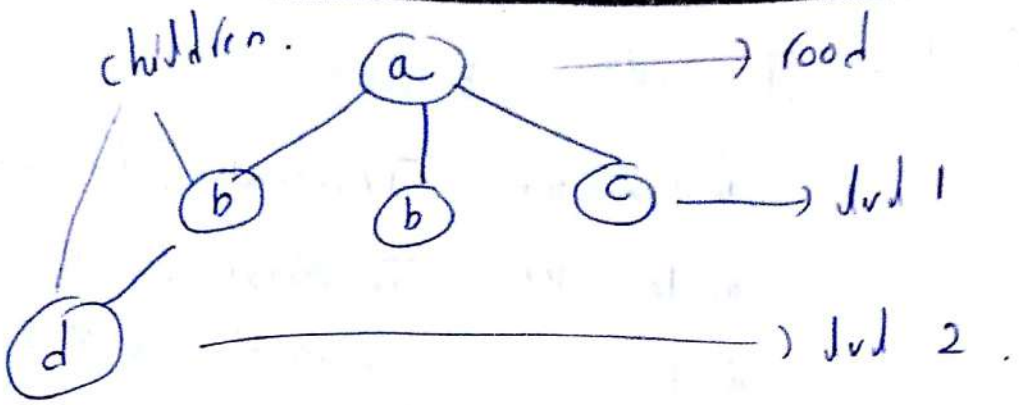
(T).

* 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

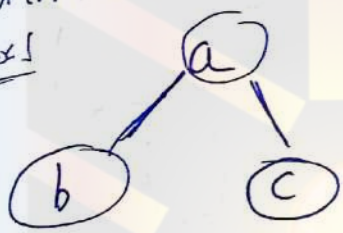


Binary tree: A tree in which there is exactly one vertex of degree two ~~& other~~

(or)

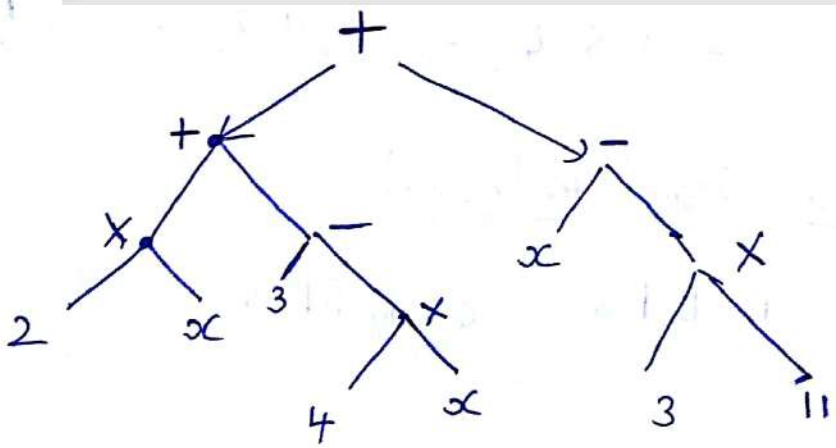
a node contain maximum 2 children.

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):

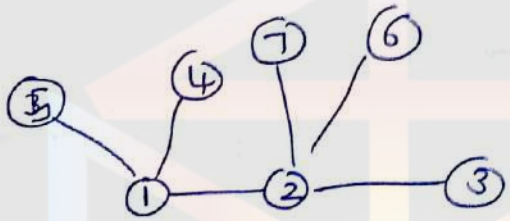
Let $G = (V, E)$ be a connected graph of order n, with vertices $v_1, v_2 \ldots v_n$ in some specified order. (Queue)

→ 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).

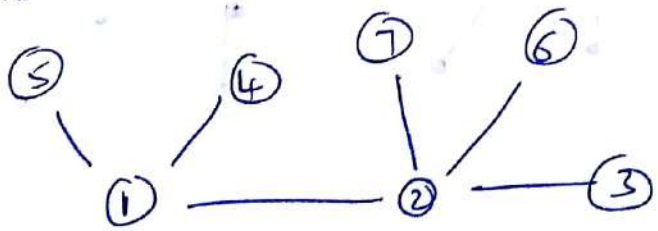
→ You can select any vertex as starting x.

→ visit one x adjacent vertex (y)

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

& continue

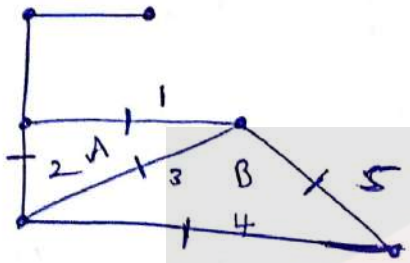
ex



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

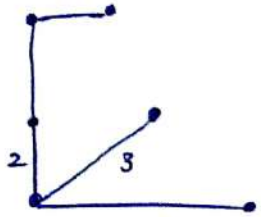


⇒ 6 vertices
&
7 edges
but,
⇒ 6 vertices ⇒ 5 edges

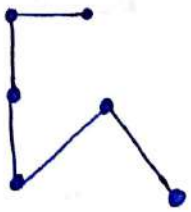
(3 × 3)
⇒ ~~3 + 3 + 3~~ − 1 ⇒ 8 possibility

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

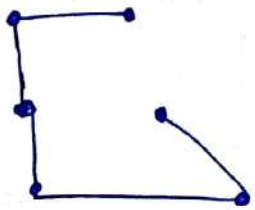
(1, 5)



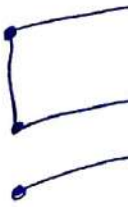
(1, 4)



(1, 3)



(2, 3



(3, 4

Mi

i)

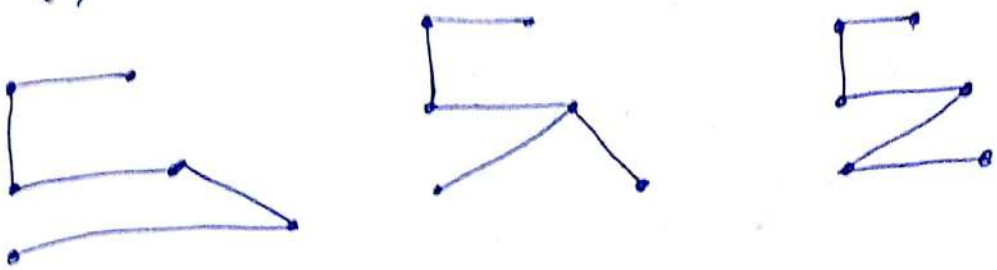
we

of

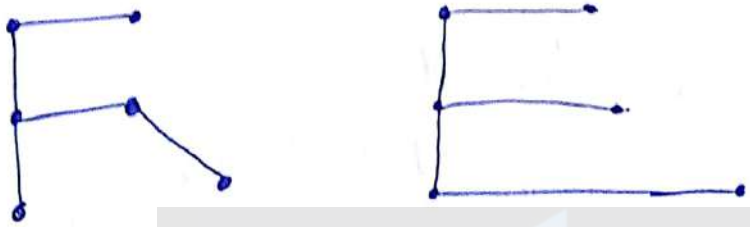
S

(

(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:

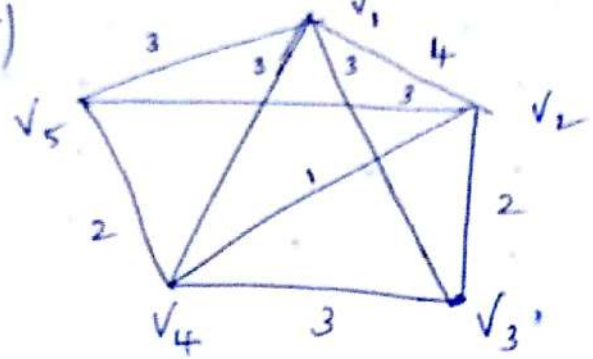
Let $G = (V, E)$ is a connected, weighted undirected graph.

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)



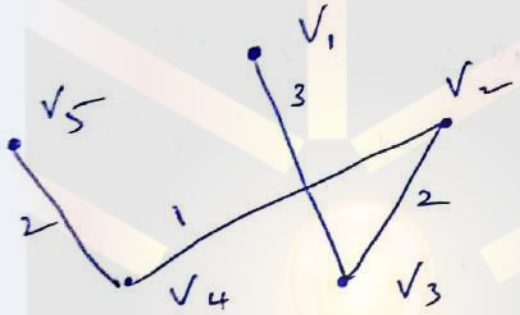
Choose V_1 search for least value

$V_1 \Rightarrow V_3$ 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 //.

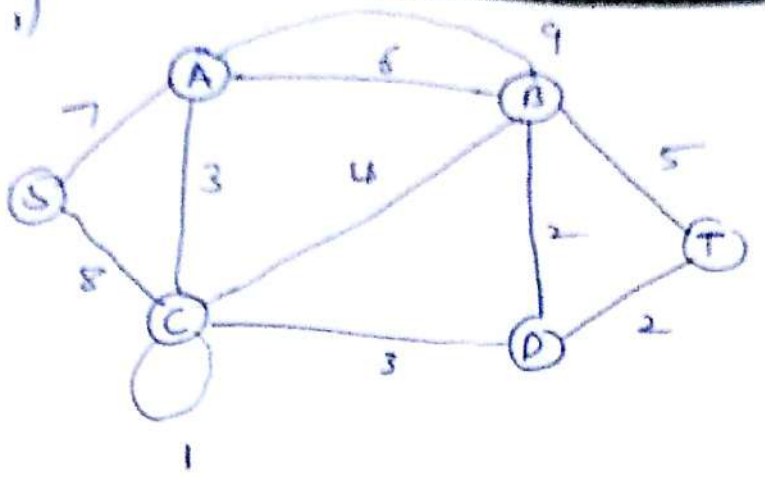
ii) Kruskal's Algorithm:

Step 1: remove all self loops & parallel edges

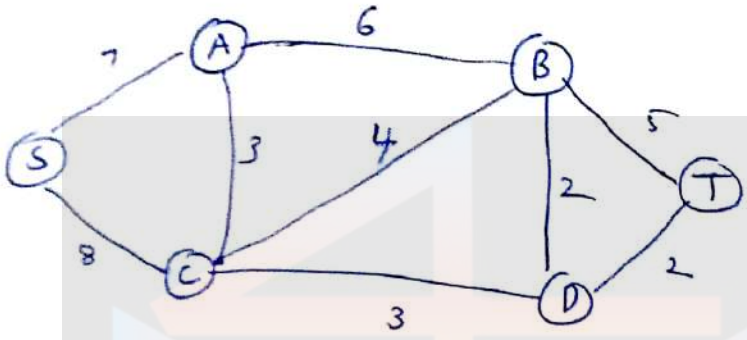
Step 2: arrange all edges in their increasing order of cost

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

Q4 1)



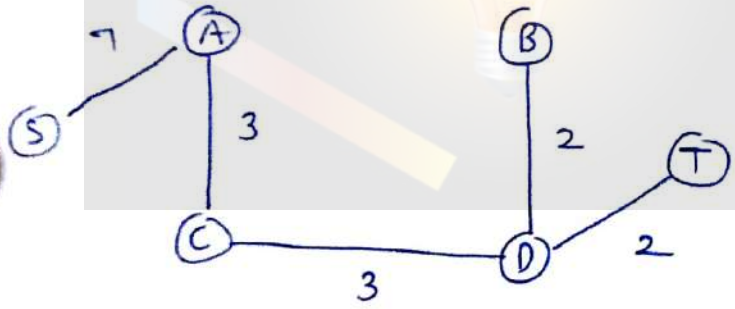
=)



=) 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.