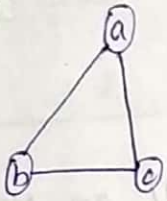
24/02/20

5. GRAPH THEORY

DEFINITION: A graph 'G' is a pair of sets [V, E] where 'V' is a non-empty set of vertices and 'E' is the pairs of vertices 'v'.

→ A Graph which is a combination of vertices and its associated edges.

Ex:

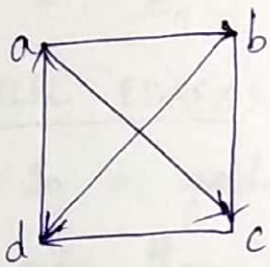


$V = \{a, b, c\}$.

$E = \{ab, bc, ca\}$.

DIRECTED GRAPH: If the graph 'G' [V, E] are represented along with its directions is called directed graph.

Ex:



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

$E = \{a-b, b-c, c-d, d-a, a-c, b-d\}$.

REPRESENTATION OF GRAPHS:-

Graphs can be represented in 3 ways.

i) Graphical representation
ii) Adjacency matrix representation
iii) Incidence matrix representation

1. GRAPHICAL REPRESENTATION:-

A graph may be represented by a diagram in which every vertex 'v' is denoted by point (or) small circle and each edge is represented by a straight line

which joins vertices.

2. ADJACENCY MATRIX REPRESENTATION:-

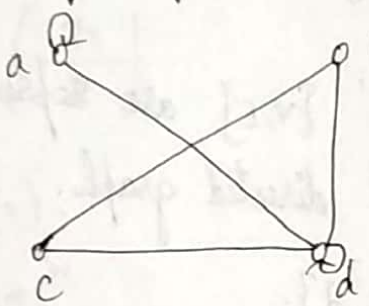
Let $G = \{V, E\}$ be a graph with 'n' vertices ordered from v_1 to v_n, an $n \times n$ matrix $A = (a_{ij})$ where

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge from } v_i \text{ to } v_j \\ 0, & \text{otherwise.} \end{cases}$$

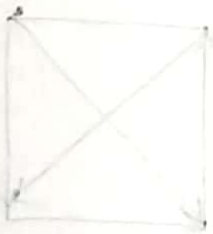
is called adjacency matrix.

*Find adjacency matrix of given graphs.

i)



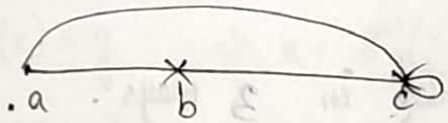
SELF-LOOP: If an vertex has edge to itself is called self-loop.



Sol:

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

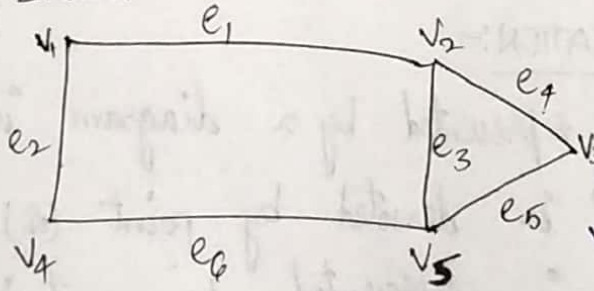
ii)



Sol:

$$A_G = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

iii) Incidence Matrix.



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

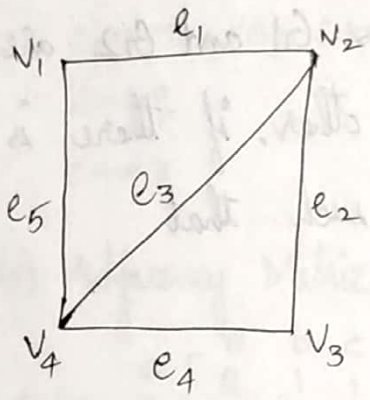
5x6

3. INCIDENCE MATRIX REPRESENTATION:-

If $G=\{V,E\}$ be a non-directed graph where $V=\{v_1, v_2, v_3, v_4, \ldots v_n\}$ and $E=\{e_1, e_2, e_3, \ldots e_m\}$ then $n \times m$ matrix 'I' $=(m_{ij})$ where

$$m_{ij} = \begin{cases} 1, & \text{if there is an incidence from } v_i \text{ to } e_j \\ 0, & \text{otherwise} \end{cases}$$

is called incidence matrix.

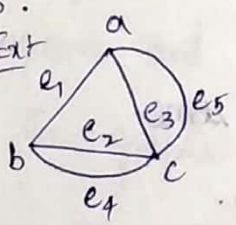
* 

$$\Rightarrow \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}_{4\times 5} \end{array}$$

PARALLEL EDGES:-

In a graph, if some pair of vertices are joined by more than one edge, such edges are called parallel edges.

Ex:-



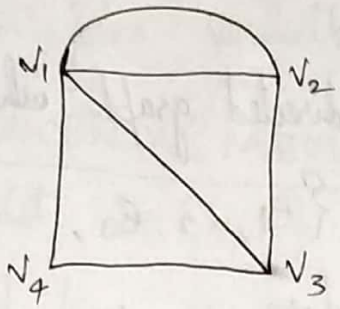
$b-c=\{e_2, e_4\}$

$a-c=\{e_3, e_5\}$.

SIMPLE GRAPH: A graph without self-loops and parallel edges is called simple graph.

DEGREE OF VERTEX: The no. of edges that are incident on the vertex of graph is called Degree of Vertex.

Ex:



$\deg(v_1) = 4$.
$\deg(v_2) = 3$
$\deg(v_3) = 3$
$\deg(v_4) = 2$.

* A vertex with degree '0' is called isolated vertex.
* A vertex with degree '1' is called pendent vertex.

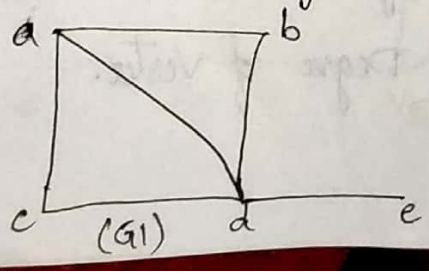
GRAPH ISOMORPHISM: Two graphs G1 and G2 are said to be isomorphic to each other, if there is a function $f: V(G1) \to V(G2)$ such that

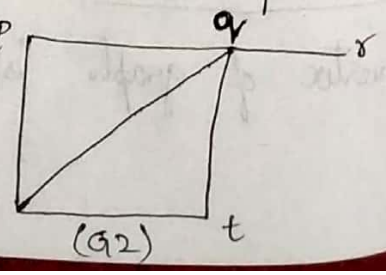
i) 'f' is one-one function
ii) 'f' is onto function
iii) 'f' preserves adjacency.

→ If two graphs are isomorphic, then

i) No. of vertices of 2 graphs are same.
ii) No. of edges are same.
iii) No. of connected components are same
iv) Adjacency matrices are also same.
v) Degree sequences are same.
vi) No. of cycles of different length are same.

*Show that the given 2 graphs are isomorphic.





Sol:- i) In G1, $|V| = 5$ In G2, $|V| = 5$. (vertices)

ii) In G1, $|E_1| = 6$ In G2, $|E_2| = 6$. (Edges)

iii) Degree sequence

a	b	c	d	e
3	2	2	4	1

G1

p	q	r	s	t
2	4	1	3	2

G2

$a \leftrightarrow s$
$b \leftrightarrow p$ (or) $b \leftrightarrow t$
$c \leftrightarrow t$ (or) $c \leftrightarrow p$
$d \leftrightarrow q$
$e \leftrightarrow r$

a	s
b	p
c	t
d	q
e	r

(or)

a	s
b	t
c	p
d	q
e	r

iv) Adjacency Matrix

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

G1

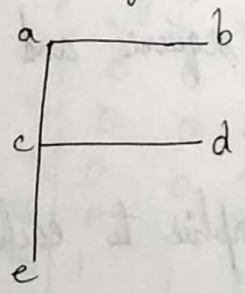
	s	p	t	q	r
s	0	1	1	1	0
p	1	0	0	1	0
t	1	0	0	1	0
q	1	1	1	0	1
r	0	0	0	1	0

G2

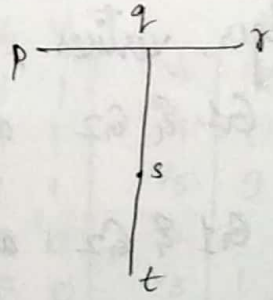
→ In graphs G1 & G2, the no. of edges, vertices, degree sequence and adjacency matrices are same.

∴ The graphs G1 & G2 are isomorphic to each other.

* Show that given 2 graphs are isomorphic.



(G1)



(G2).

Sol: i) Vertices

In G_1, $|V| = 5$ In G_2, $|V| = 5$

ii) Edges

In G_1; $|E| = 4$ In G_2, $|E| = 4$.

iii) Degree sequence

a	b	c	d	e
2	1	3	1	1

G_1

p	q	r	s	t
1	3	1	2	1

G_2

$a \leftrightarrow s$

$b \leftrightarrow p$ (or) $b \leftrightarrow r$ (or) $b \leftrightarrow t$

$c \leftrightarrow q$

$d \leftrightarrow r$ (or) $d \leftrightarrow t$ (or) $\leftrightarrow d \leftrightarrow p$

$e \leftrightarrow t$ (or) $e \leftrightarrow p$ (or) $e \leftrightarrow r$

a	b	c	d	e
s	p	q	r	t

(or)

a	b	c	d	e
s	r	q	t	p

(or)

a	b	c	d	e
s	t	q	p	r

iv) Adjacency matrix

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

(G_1)

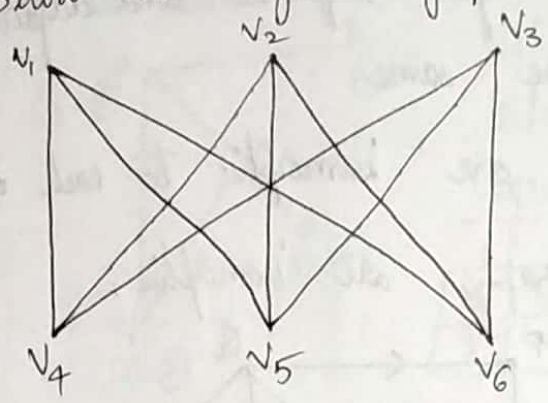
	s	t	q	p	r
s	0	1	1	0	0
t	1	0	0	0	0
q	1	0	0	1	1
p	0	0	1	0	0
r	0	0	1	0	0

(G_2)

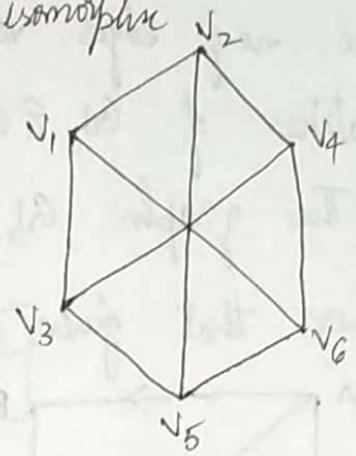
→ The no. of edges, vertices, degree sequence and adjac matrices of G_1 & G_2 are same.

∴ The graphs G_1 & G_2 are isomorphic to each oth

* Show that given graphs are isomorphic



(G1)



(G2)

Sol:- i) Vertices

In G1, $|V| = 6$ In G2, $|V| = 6$.

ii) Edges

In G1, $|E| = 9$ In G2, $|E| = 9$.

iii) Degree sequence.

v_1	v_2	v_3	v_4	v_5	v_6
3	3	3	3	3	3

(G1)

v_1	v_2	v_3	v_4	v_5	v_6
3	3	3	3	3	3

(G2)

$v_1 \leftrightarrow v_1$
$v_2 \leftrightarrow v_2$
$v_3 \leftrightarrow v_3$
$v_4 \leftrightarrow v_4$
$v_5 \leftrightarrow v_5$
$v_6 \leftrightarrow v_6$

v_1	v_1
v_2	v_2
v_3	v_3
v_4	v_4
v_5	v_5
v_6	v_6

iv) Adjacency Matrix

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

(G1)

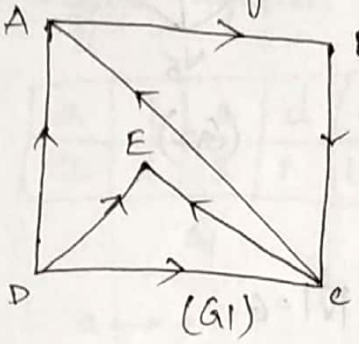
	v_1	v_4	v_5	v_2	v_3	v_6
v_1	0	0	0	1	1	1
v_4	0	0	0	1	1	1
v_5	0	0	0	1	1	1
v_2	1	1	1	0	0	0
v_3	1	1	1	0	0	0
v_6	1	1	1	0	0	0

(G2)

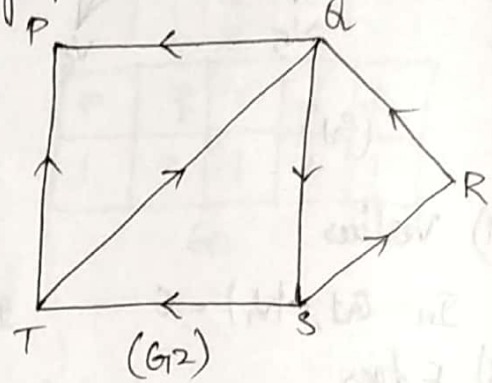
→ The no. of edges, vertices, degree sequence and adjacency matrices of G1 & G2 are same.

∴ The graphs G1 & G2 are isomorphic to each other.

* Show that given two graphs are isomorphic.



(G1)



(G2)

Sol:- i) Vertices

In G1, $|V| = 5$ In G2, $|V| = 5$

ii) Edges

In G1, $|E| = 7$ In G2, $|E| = 7$.

iii) Degree sequence

	A	B	C	D	E
In degree	2	1	2	0	2
Out degree	1	1	2	3	0

(G1)

	P	Q	R	S	T
In degree	2	2	1	1	1
Out degree	0	2	1	2	2

(G2)

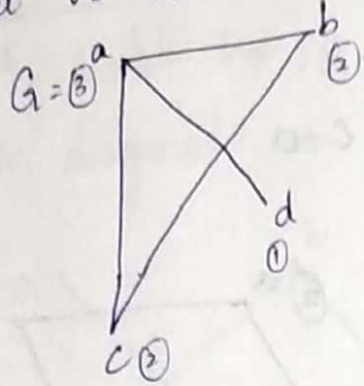
→ In G1 & G2, the degree sequences are not same.

∴ The given graphs G1 & G2 are not isomorphic to each other.

SUB GRAPH:-

→ Let $G = \{V, E\}$ be a graph, any sub part of the given graph 'G' is called sub-graph of the given graph.

→ Let us consider



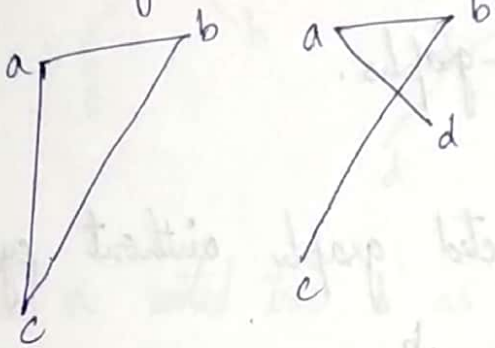
$|V| = 4$.

$|E| = 4$

Degree sequence

a	b	c	d
3	2	2	1

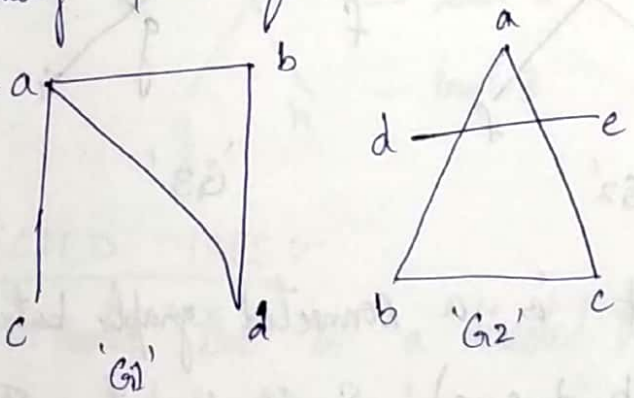
It's sub-graphs are



CONNECTED GRAPH:-

→ A graph 'G' is connected, if there is a 'tour' b/w every pair of its vertices.

Ex:-



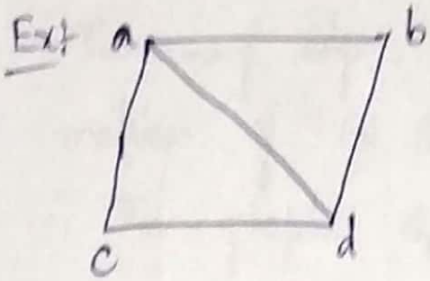
⇒ 'G2' is not connected graph, because there is no direct (or) indirect edge from 'a-d'.

CYCLIC GRAPH:-

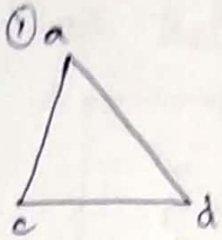
→ A circular 'tour' through different vertices & edges is called cyclic.

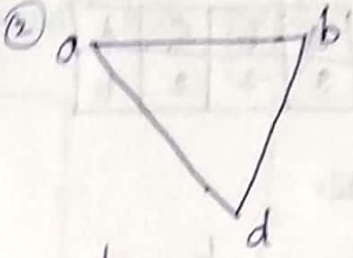
→ Starting vertex and ending vertex is same.

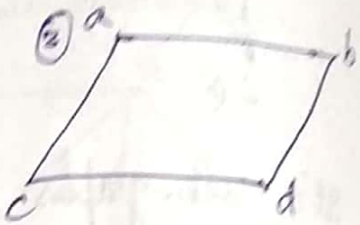
Ex:



① a-d-c-a.
② a-b-d-a.
③ a-b-d-c-a.



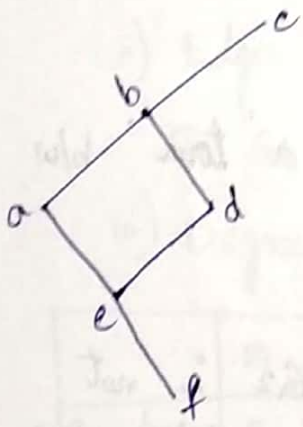




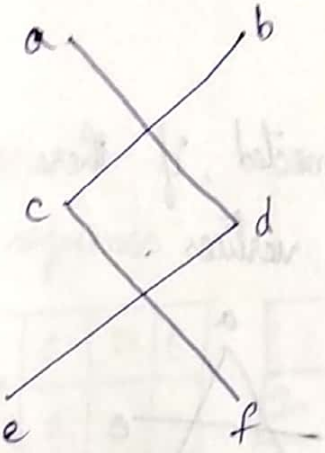
* Cyclic graphs are also sub-graphs.

TREE:-

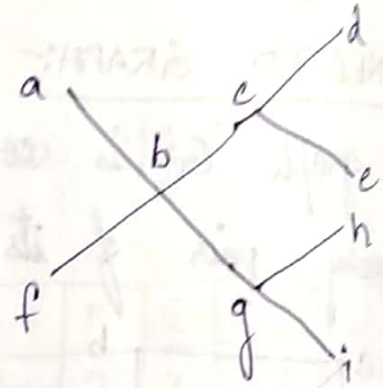
→ A tree 'T' is a connected graph without cycles.



'G1'



'G2'



'G3'

→ In graph 'G1', it is a connected graph but it has cycle (a-b-d-e-a), so it is not a Tree.

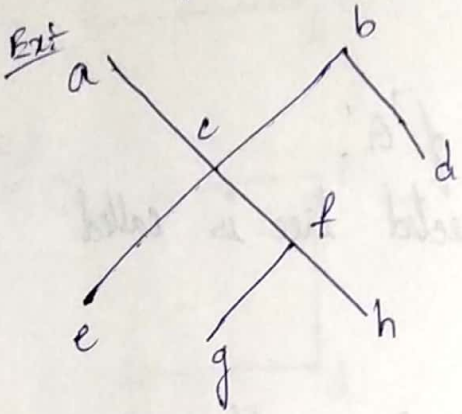
→ In graph 'G2', it is not connected graph, so it is not a Tree.

→ In graph 'G3', it is a connected graph and it has no cycles, so it is a Tree.

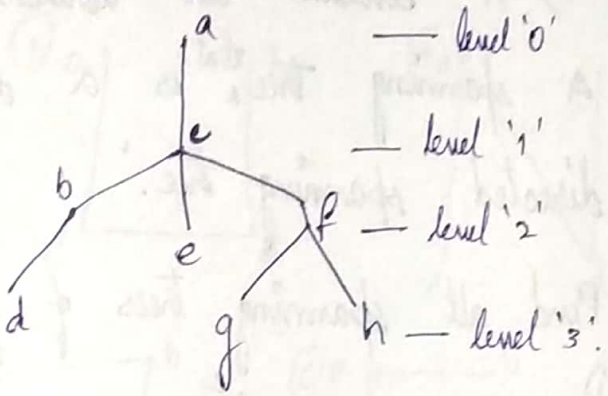
ROOTED TREE:-

→ A rooted tree is a tree in which a particular vertex is designated as a root.

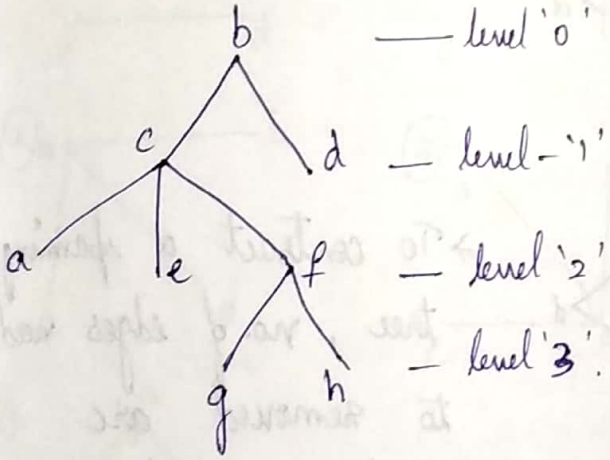
Ex:-



⇒ Construct a rooted tree root as 'a'.



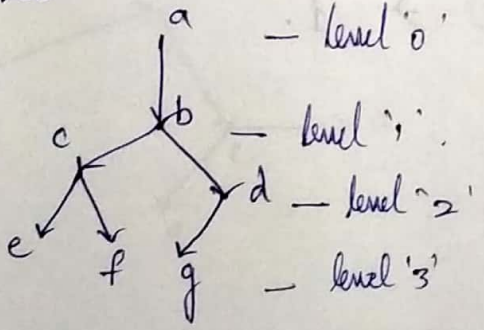
⇒ Construct a rooted tree 'b' as root.



DIRECTED TREE:-

→ A rooted tree is a directed tree, if there is a root from which there is a directed path to each vertex.

Ex:-



SPANNING TREE:-

→ A sub-graph 'H' of graph 'G' is called a spanning tree of 'G', if

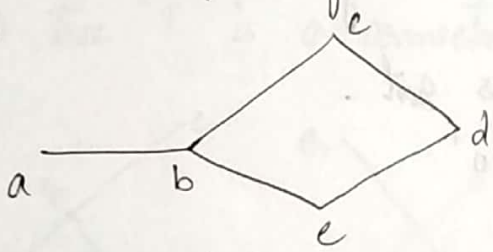
i) 'H' is a tree.

ii) 'H' contains all vertices of 'G'.

→ A spanning tree that is a directed tree is called 'directed spanning tree.'

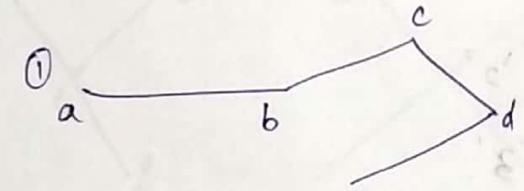
* Find all spanning trees of following graphs.

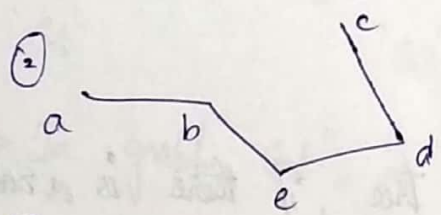
i)

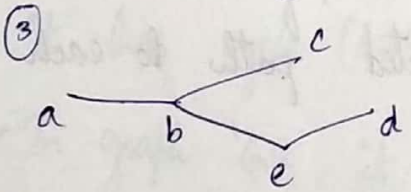


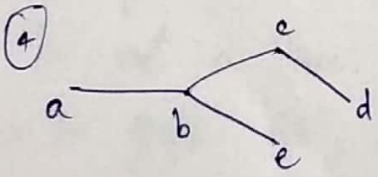
'G'.

Sol:









→ To construct a spanning tree, no. of edges need to removed are

$|V| = 5 \quad |E| = 5.$

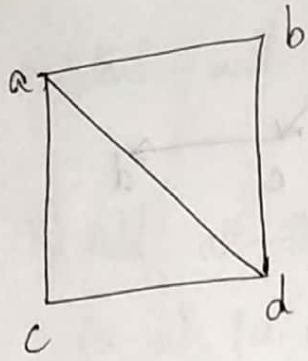
$\Rightarrow |E| - |V| - 1|$

$\Rightarrow 5 - |5-1|$

$\Rightarrow 5 - 4$

$\Rightarrow 1.$

ii)



⇒ No. of edges need to be removed are:

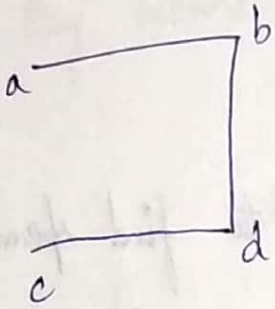
⇒ $|E| - |V-1|$

⇒ $5 - |4-1|$

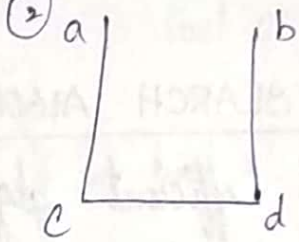
⇒ $5 - 3 = 2$.

Sol:

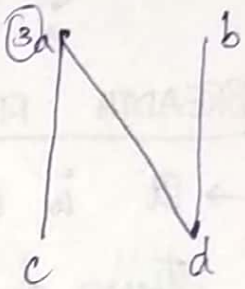
①



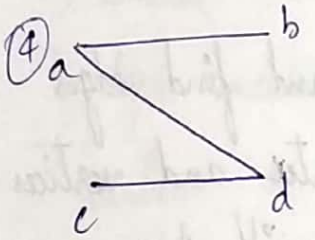
②



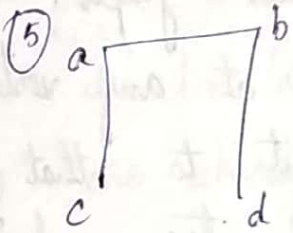
③



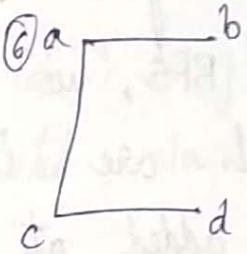
④



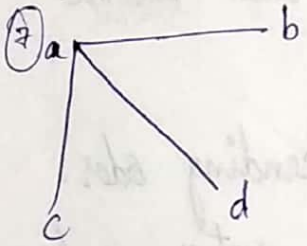
⑤



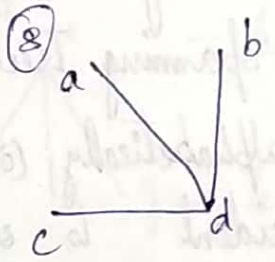
⑥



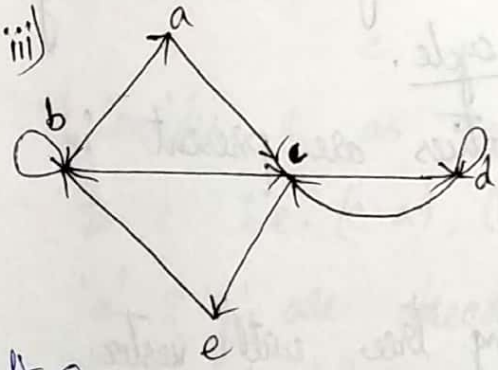
⑦



⑧



iii)

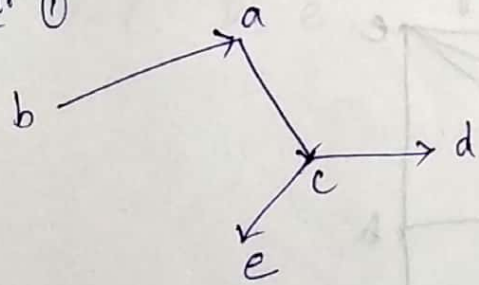


⇒ No. of edges need to be removed are $|E| - |V-1|$

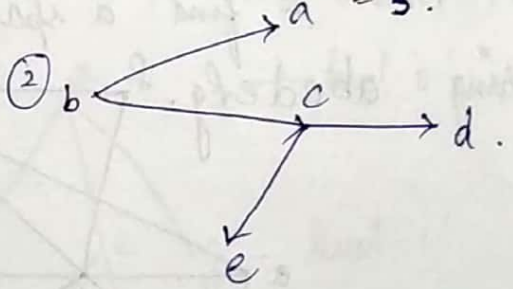
$= 9 - |5-1|$

$= 5$.

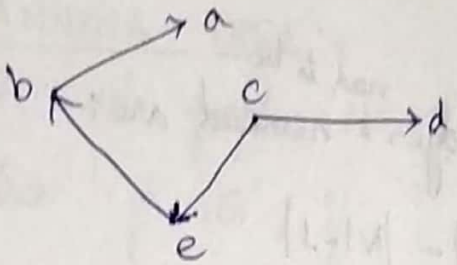
Sol: ①



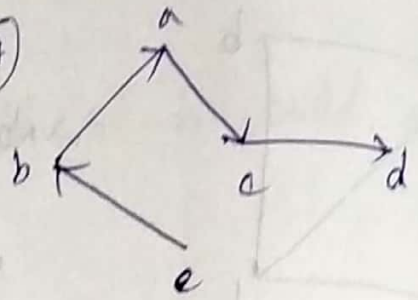
②



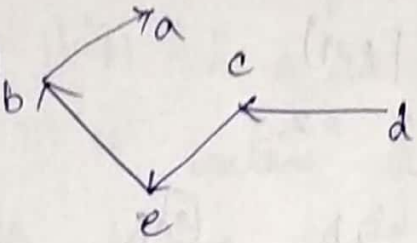
(3)



(4)



(5)



1. BREADTH FIRST SEARCH ALGORITHM:-

→ It is one of efficient algorithm to find spanning tree of a connected graph.

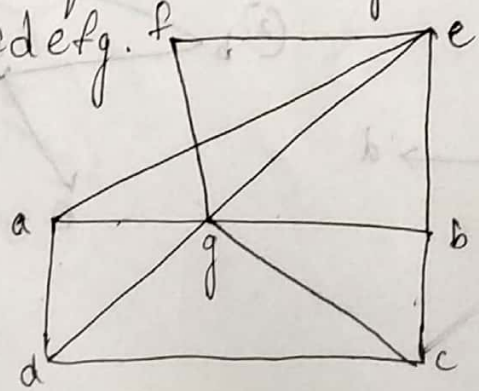
→ In BFS, we start at any vertex and find edges which are incident to that vertex and vertices are added at each stage and it will become level-1 vertices in spanning tree.

→ We add vertices alphabetically (or) ascending order.

→ We add edges incident to every vertex as long as it doesn't produce a cycle.

→ Repeat step-'4' until no vertices are present in the given graph.

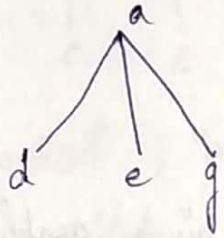
* Use BFS to find a spanning tree with vertex ordering abcdefg.



Sol: i) Start with vertex 'a' as root node.

'a'

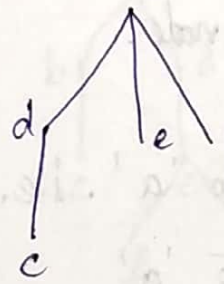
ii) Add all the edges incident to 'a', i.e. (a-d), (a-e), (a-g). Add d, e, g vertices to 'a', they become level-one vertices.



– level '0'

– level '1'.

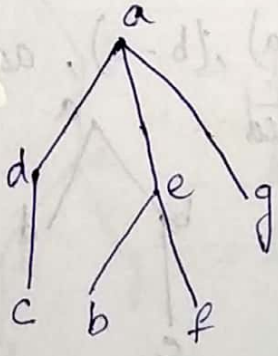
iii) Consider 'd' as root in level-1 and add the edges incident to 'd' i.e. (d-a), (d-c), (d-g). The vertices 'a' & 'g' are already visited, Now add vertex 'c' to 'd'.



– level '0'

– level-1'

iv) Consider 'e' as root in level '1', add edges incident to 'e' i.e. (e-a), (e-b), (e-f), (e-g). The vertices 'a' & 'g' are already visited, now add vertices 'b', 'f' to 'e'.



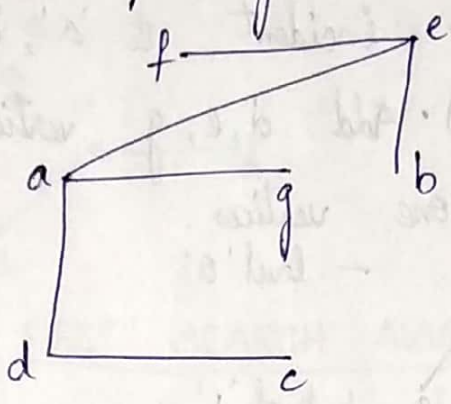
– level '0'

– level '1'

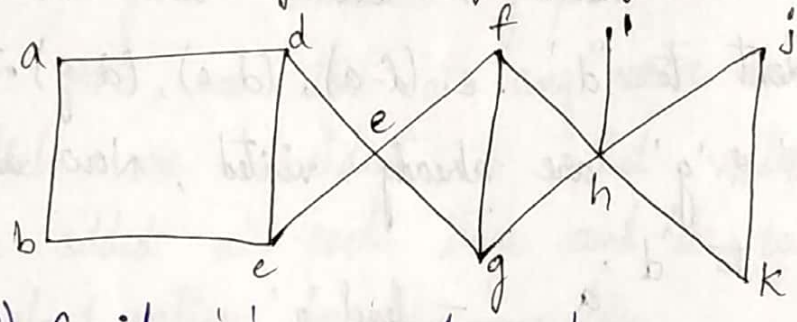
– level '2'.

→ This is the required graph in BFS algorithm.

→ The BFS spanning tree according to given graph is



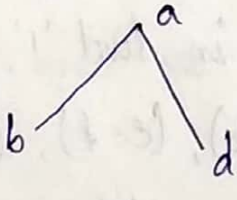
* Draw spanning tree of given graph using BFS algo



Sol:- i) Consider 'a' as root node.

'a'

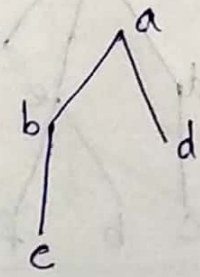
ii) Add edges incident to 'a'. i.e. (a-b), (a-d).
add b, d vertices to 'a'.



a - level '0'

d - level '1'

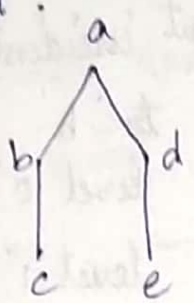
iii) Consider 'b' as root, add vertices incident to b, they are (b-a), (b-c), add 'c' to 'b' as 'a' is already visited.



a - level '0'

d - level '1'

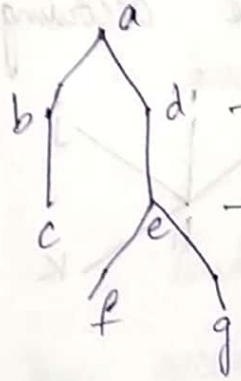
iv) Consider 'd' as root, add vertices incident to 'd' i.e. a, c, e. Now add 'e' to 'd' as a, c are already visited.



a – level '0'
d – level '1'
c, e – level '2'.

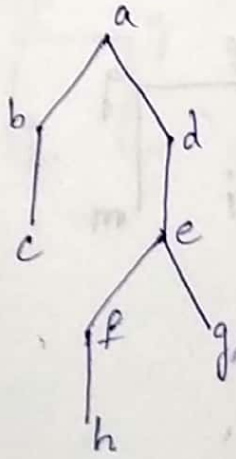
v) Consider 'c' as root, but its incident vertices are already visited.

vi) Consider 'e' as root, it's incident vertices are c, d, f, g. 'c', 'd' are already visited, so add vertices f, g to 'e'.



a – level '0'
d – level '1'
e – level '2'.
f, g – level '3'.

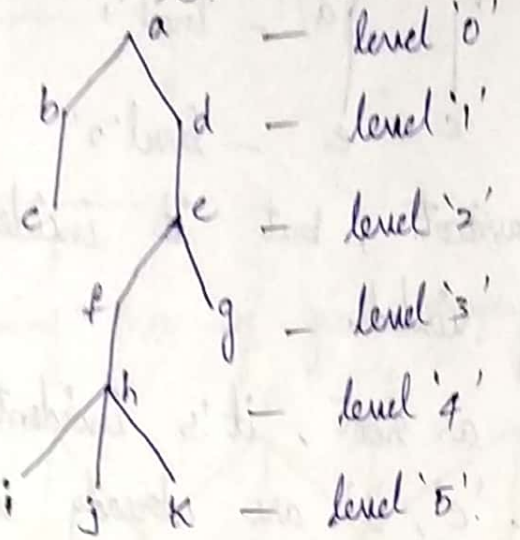
vii) Consider 'f' as root, its incident vertices are e, g, h. Add 'h' to 'f' as e, g are already visited



a – level '0'
d – level '1'
e – level '2'
f, g, h – level '3'.

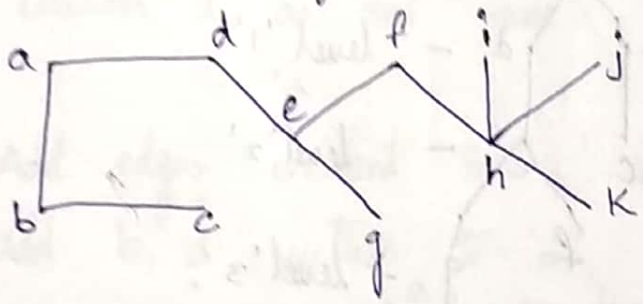
viii) Consider 'g' as root, but its incident vertices are already visited.

ix) Consider 'h' as root, its incident vertices are i, j, K. Add i, j, K to 'h'.

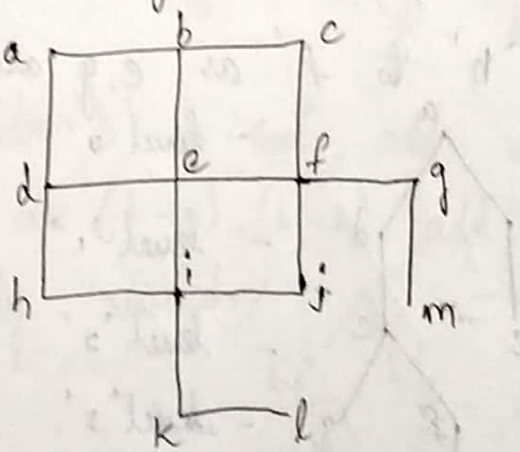


→ This is the required graph in BFS algorithm.

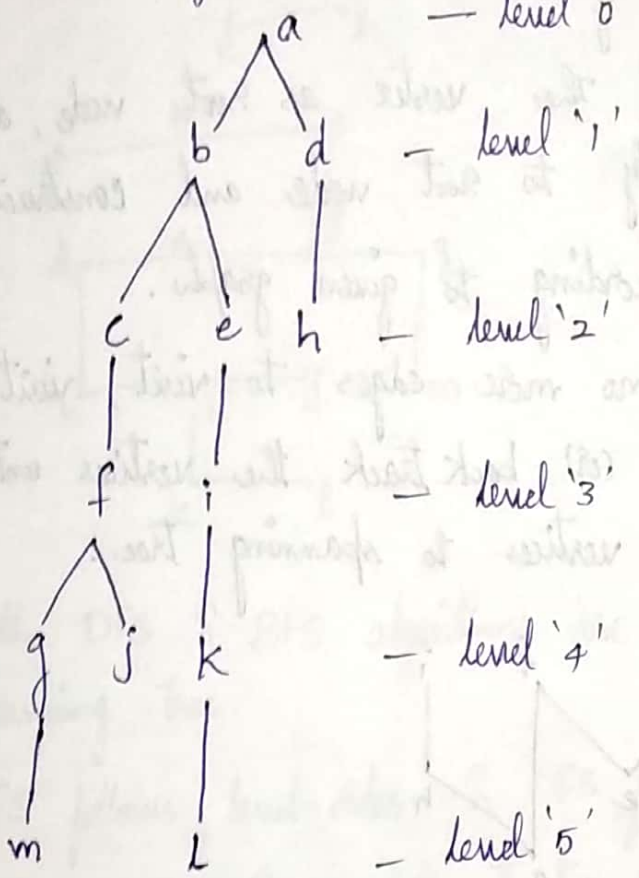
→ The BFS spanning tree according to given graph is



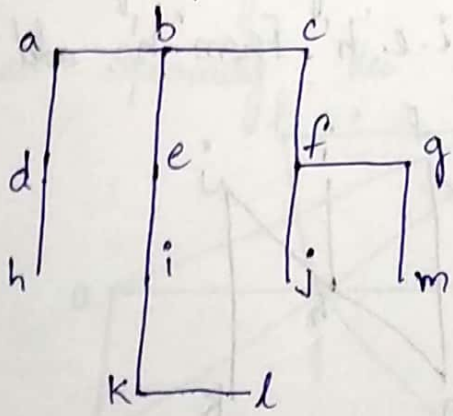
* Apply BFS algorithm and draw spanning tree.



Sol:→ Consider 'a' as root and add vertices which are adjacent to it. Make successive vertices as root and keep on add vertices to tree until all the vertices gets visited.

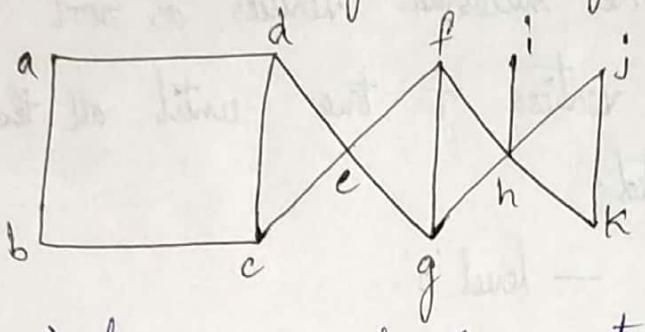


→ The spanning tree according to BFS graph is



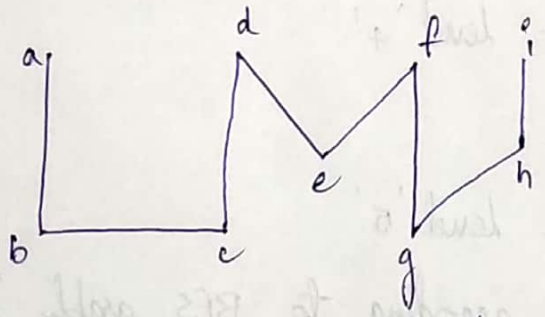
2. DEPTH FIRST SEARCH ALGORITHM:-

* Construct spanning tree using DFS algorithm.

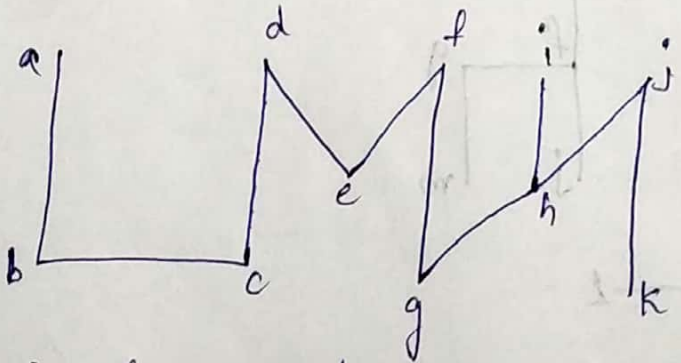


Sol: i) choose one of the vertex as root node, add edges successively to root node and construct DFS tree according to given graph.

ii) If there are no more edges to visit, visit its previous vertex (or) back track the vertices until we add all vertices to spanning tree.

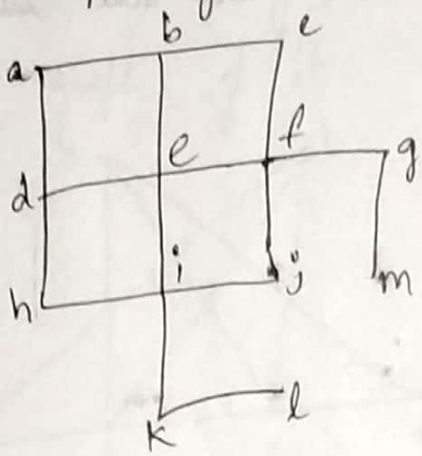


→ At 'i' we can't add any more edges, so visit its previous vertex i.e. 'h'. From 'h', add 'j' & 'j-k'.

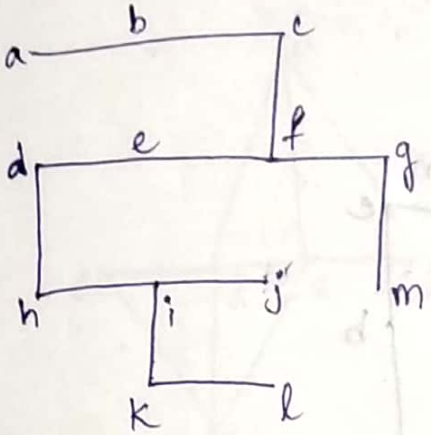


→ It is the required DFS spanning tree from given graph.

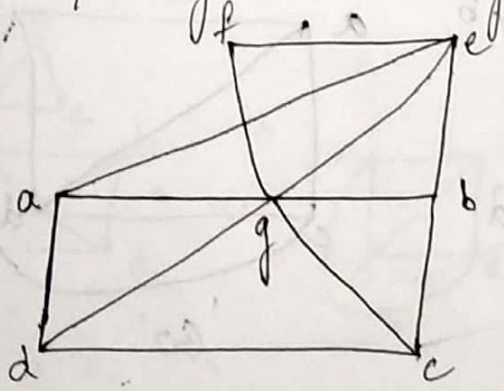
* Construct spanning tree using DFS algorithm.



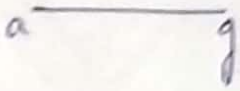
Sol:



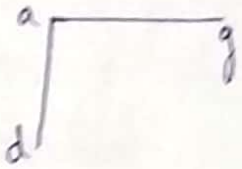
* Both DFS & BFS algorithms are used to construct Spanning tree.
* BFS follows level-order & DFS follows pre-order.
* BFS uses Queue data structure & DFS uses Stack.
* No Backtracking in BFS, Backtracking is used in DFS.
* Construct Spanning tree using DFS algorithm.



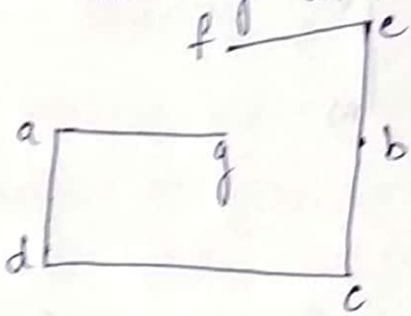
8d:- i) Consider 'g' as root node and add incoming vertices to it.



ii) Consider 'a' as root.



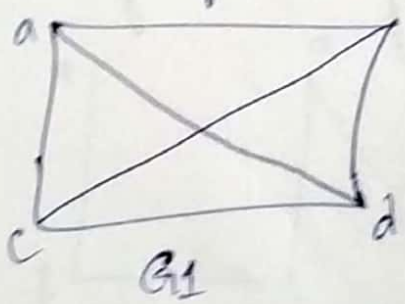
iii) Then 'd' and 'g' on



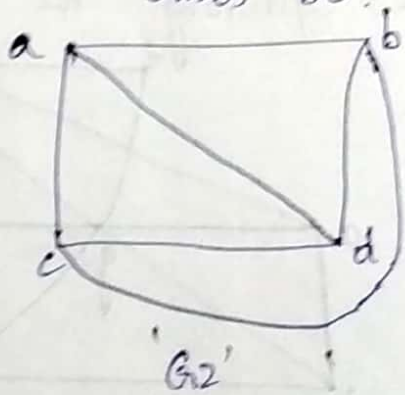
PLANAR GRAPHS:-

→ A graph 'G' is said to be planar, if it can be drawn in a plane, so that it edges doesn't intersect (or) cross over.

→ Graph 'G1' contains 4 vertices & 6 edges, it is not a planar graph, because, 'ad' crosses 'bc'.



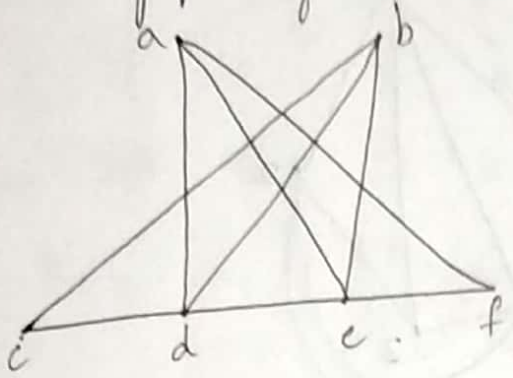
G1



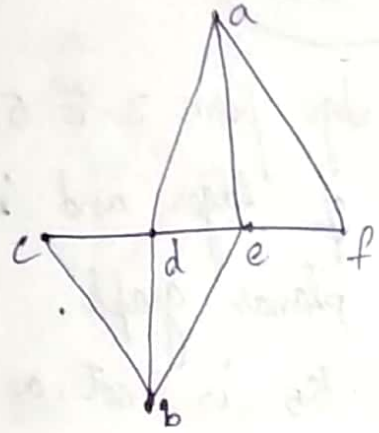
G2

→ G2 is planar.

* Show that given graph is planar, if not draw the planar graph of it.



Sol:



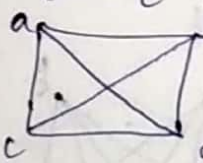
COMPLETE GRAPH: A graph with 'n' vertices, so that each of 'n' vertices are adjacent to other each 'n-1' vertices. is called complete graph.

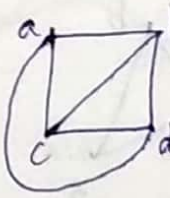
→ It is denoted by "K_n".

⇒ K_1 a

⇒ K_2 

⇒ K_3 

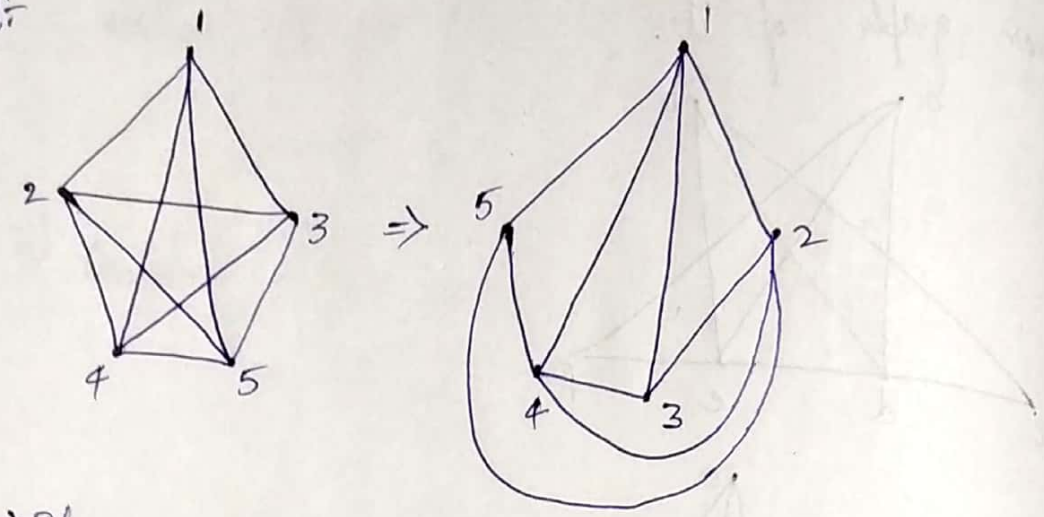
⇒ K_4 

⇒ 

(Planar graph).

* show that K_5 is not a planar graph.

Sol:



⇒ If we want to draw edge from 3 to 5, it must interset any one of edges and it violates the basic condition of planar graph.

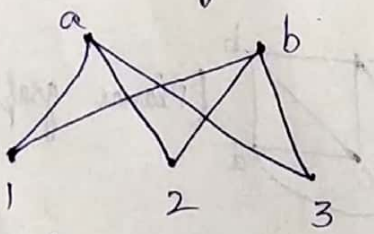
⇒ So, the complete graph K_5 is not a planar graph.

COMPLETE BIPARTITE GRAPH: A graph with 'm+n' vertices, so that each of first 'm' vertices are adjacent to each of other 'n' vertices.

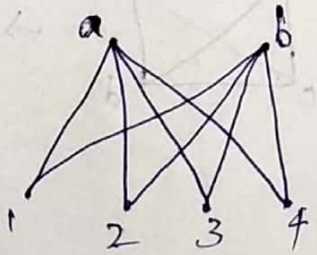
→ There is no edge b/w first 'm' vertices and also in second 'n' vertices.

→ It is denoted by $K_{m,n}$.

⇒ $K_{2,3}$



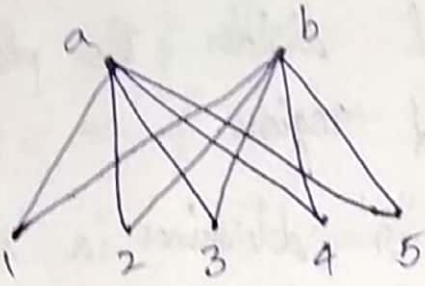
⇒ $K_{2,4}$



⇒ m+n ⇒ vertices

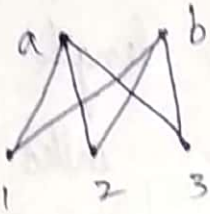
⇒ m*n ⇒ edges.

⇒ $K_{2,5}$

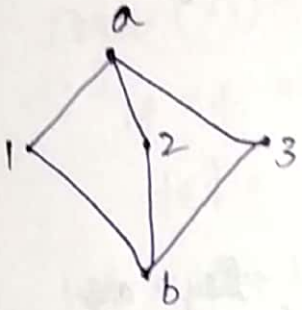


* Show that $K_{2,3}$ is a planar graph.

Sol: $K_{2,3}$ =

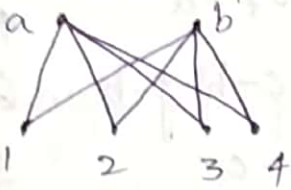


⇒

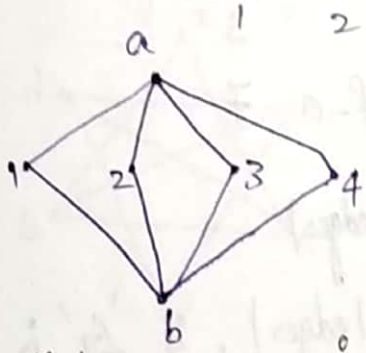


* Show that $K_{2,4}$ is a planar graph.

Sol: $K_{2,4}$ =

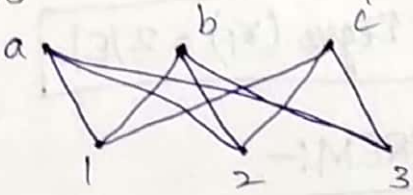


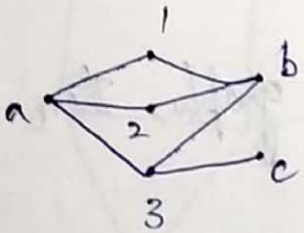
⇒



* Show that $K_{3,3}$ is not a planar graph.

Sol: $K_{3,3}$ ⇒





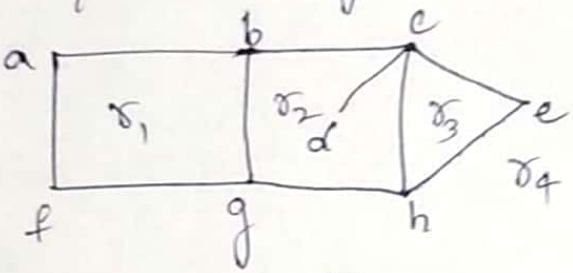
⇒ We can't draw c-1 without intersecting other edges.

REGION: A planar graph divides the plane into connected areas called regions.

→ Each plane graph 'G' determines a region of infinite area called exterior region of 'G'.

DEGREE OF REGION: The length of the closed path of a particular region is called degree of a region.

Ex:-



⇒ given graph contains 4 regions, they are

	Path	length
r_1 →	a-b-g-f-a	4
r_2 →	b-c-d-c-h-g-b	6
r_3 →	c-e-h-c	3
r_4 →	a-b-c-e-h-g-f-a	7

⇒ degree $(r_i) = 20 = 2 \times |edges|$

* Degree of region is $2 \times |edges|$, if 'G' is a planar graph.

$$\boxed{\text{Degree}(r_i) = 2|E|.}$$

EULAR'S THEOREM:-

→ If 'G' is a connected plane graph, then

$$\boxed{|V| - |E| + |R| = 2}$$

where, V = no. of vertices

E = no. of edges.

R = no. of regions.

* Suppose 'G' is a connected planar graph with 14 regions each of its degree '4'. Find the no. of vertices in 'G'.

Sol: Given, $|R| = 14$

Degree $(r_i) = 2|E|$.

$14 \times 4 = 2 \times |E|$.

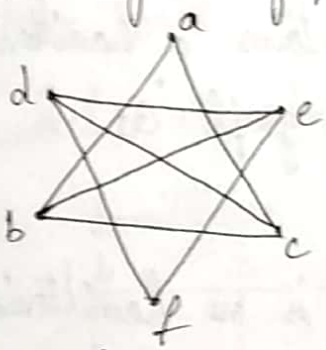
$|E| = 28$.

$\therefore |V| = 2 + |E| - |R|$.

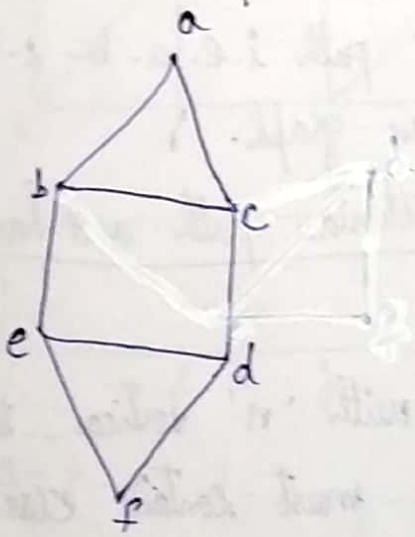
$= 2 + 28 - 14$

$= 16$.

* Show that given graph is planar.



Sol:

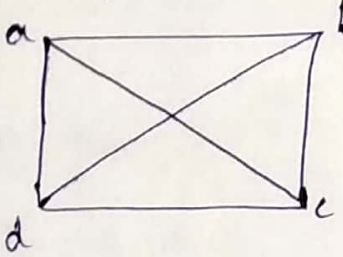


HAMILTONIAN GRAPHS: A graph 'G' is said to be hamiltonian graph, if it contains hamiltonian cycle.

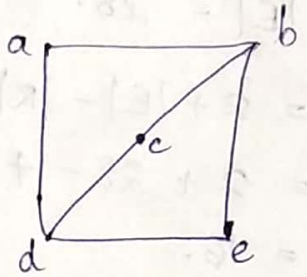
→ HAMILTONIAN CYCLE: A cycle in a graph 'G' is called hamiltonian cycle, if it contains every vertex of 'G'.

→ HAMILTONIAN PATH: A path in a graph 'G' is called hamiltonian path, if it contains every vertex of 'G'.

Ex:-



'G1'



'G2'.

→ In first graph 'G1', it contains a hamiltonian cycle i.e. a-b-c-d-a, So the graph 'G1' is a hamiltonian graph.

→ In second graph 'G2', there is no hamiltonian cycle, but it has a hamiltonian path i.e. a-b-c-d-e, So it is not a hamiltonian graph.

Rules for Constructing hamiltonian path and hamiltonian cycle in a Graph 'G':-

Rule-1: If 'G' is a graph with 'n' vertices, then a hamiltonian cycle must contain exactly 'n'

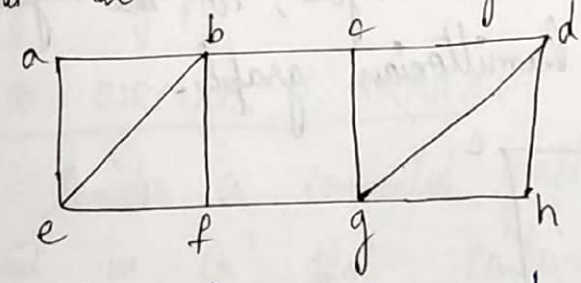
edges & hamiltonian path must contain 'n-1' edges.

Rule-2: If 'v' is a vertex of degree '2', then both edges incident on 'v' should contain in every hamiltonian cycle and every hamiltonian path should contain atleast one edge incident on 'v'.

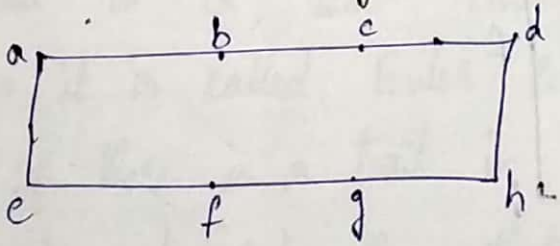
Rule-3: No cycle that doesn't contain all vertices of 'G' can be formed when building a hamiltonian path (or) cycle.

Rule-4: Once a hamiltonian cycle is constructed from vertex 'v' then its unused edges which are incident to 'v' can be deleted.

* Find the hamiltonian cycle in the following graph.

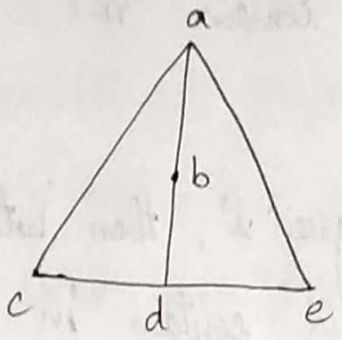


Sol:



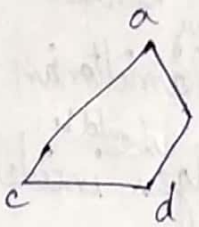
⇒ a-b-c-d-h-g-f-e-a.

*



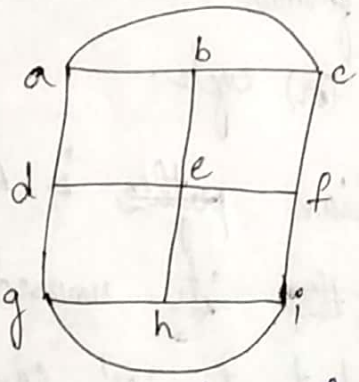
⇒ It is not a hamiltonian graph, because there is no hamiltonian cycle. Hamiltonian path is b-a-c-d-e.

Sol:

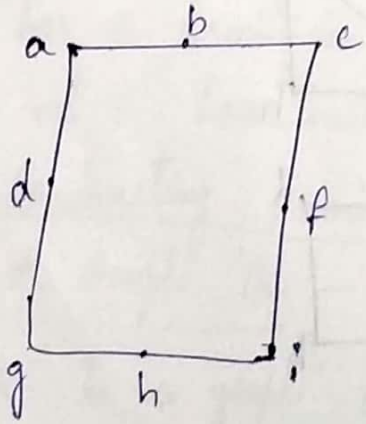


⇒ It doesn't have all vertices.
⇒ Rule-2 got violated.

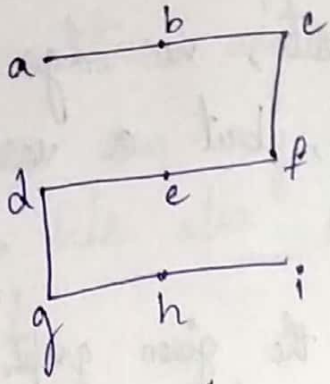
*Show that the following graph has no hamiltonian cycle, check whether graph contains any hamiltonian path.



Sol: While constructing hamiltonian cycle, vertex 'e' is not present in the cycle, so the given graph is not a hamiltonian graph.

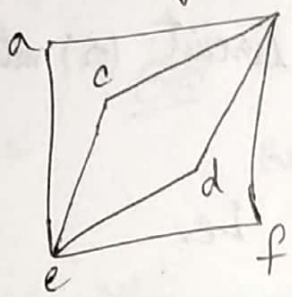


⇒ Hamiltonian path: a-b-c-f-e-d-g-h-i



* check whether given graph is hamiltonian (or) not.

'G' =



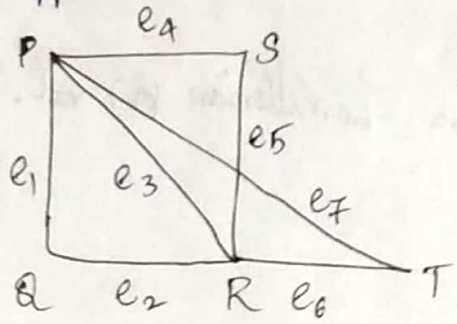
Sol: Given graph 'G' is not a hamiltonian graph since we have '6' vertices & '8' edges out of 4 vertices of degree '2'. According to rule '2', 8 edges must be included in every hamiltonian cycle. If '8' edges are present in hamiltonian cycle of 4 vertices, it violates rule-1. So, the given graph 'G' is not a hamiltonian graph.

EULER CIRCUITS & TRAILS:-

Consider a connected graph 'G', if there is a circuit in 'G', that contains all edges of 'G', then it is called Euler circuit.

→ If there is a trail in 'G', that contains all edges of 'G', then that trail is called Euler trail.

NOTE: In Euler trail (or) circuit, no edge can appear more than once, but a vertex can appear more than once.

* 

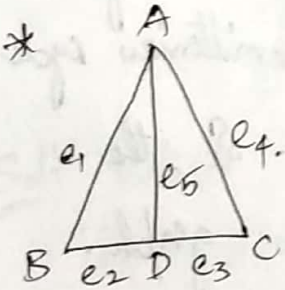
Consider the given graph 'G' and check whether it contains euler circuit (or) not.

Sol: 'G' contains Euler's circuit i.e.

P e1 Q e2 R e3 P e7 T e6 R e5 S e4 P

→ No edge is repeated in given sequence, so the given graph 'G' ~~contains~~ euler's circuits.

⇒ P e1 Q e2 R e6 T e7 P e3 R e5 S e4 P.

* 

Sol: A e1 B e2 D e3 C e4 A e5 D.

The given graph is not a euler circuit because while constructing the circuit, the edges are getting repeated more than once.

But, it has euler trail (or) euler path i.e.

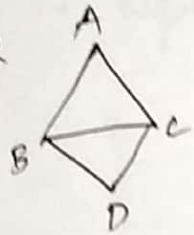
A e1 B e2 D e5 A e4 C e3 D.

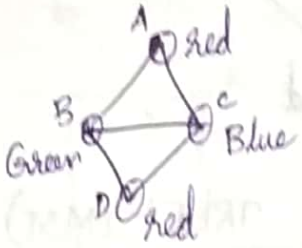
CHROMATIC NUMBER (OR) GRAPH COLOURING:-

Given a planar (or) non planar graph 'G', if we assign colour to its vertices in such a way

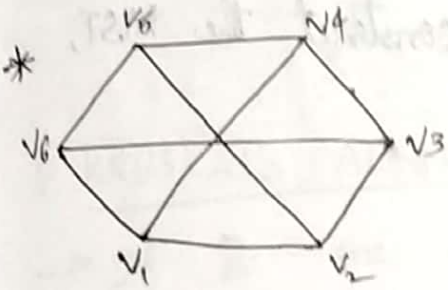
that no two adjacent vertices have same colour, then we can say that graph is properly coloured. We also call it as chromatic number of graph.

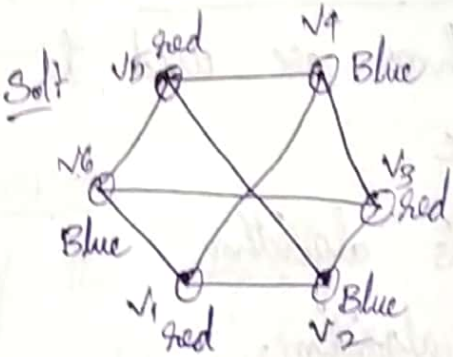
→ It is denoted as $\chi(G)$.

* 

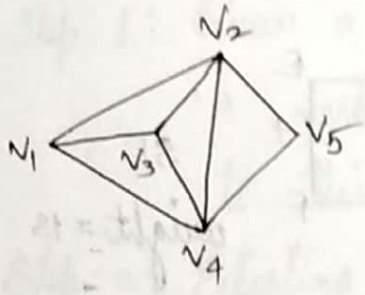
Soln 

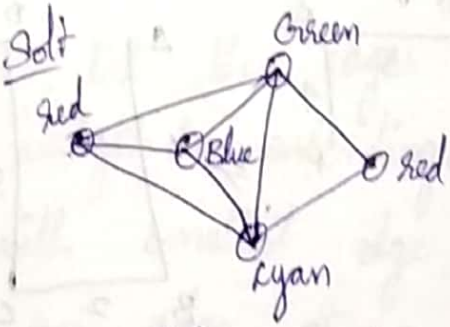
$\chi(G) = 3$.

* 

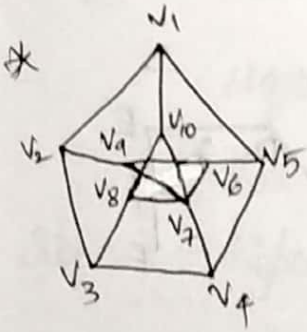
Soln 

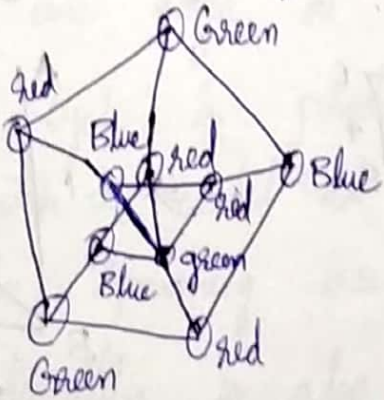
$\chi(G) = 2$.

* 

Soln 

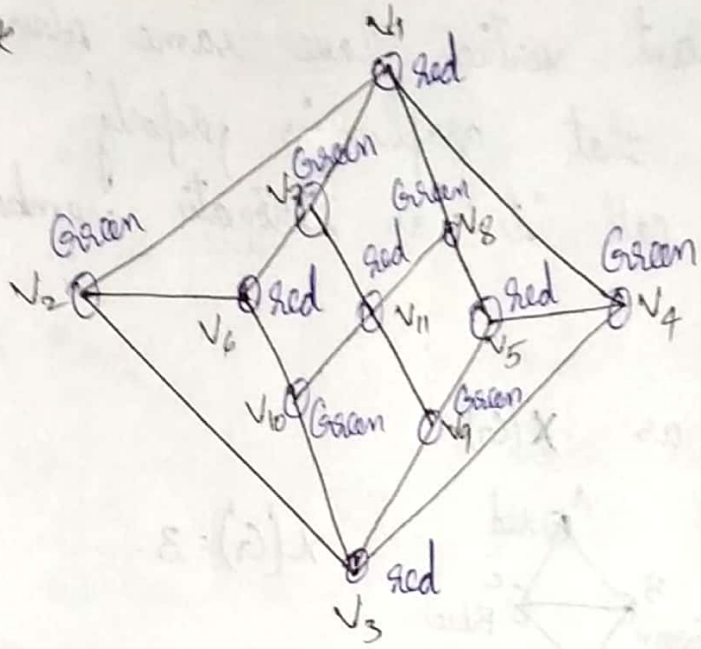
$\chi(G) = 4$.

* 

Soln 

$\chi(G) = 3$.

*



Red - V_1, V_6,

$\chi(G) = 2$.

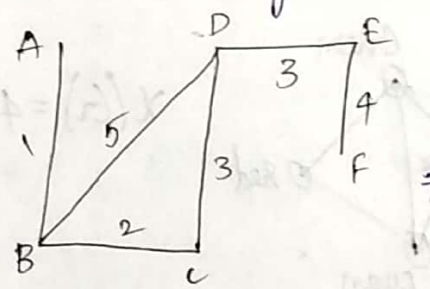
MINIMAL SPANNING TREE:- (MST).

→ Two algorithms are used to construct the MST, they are

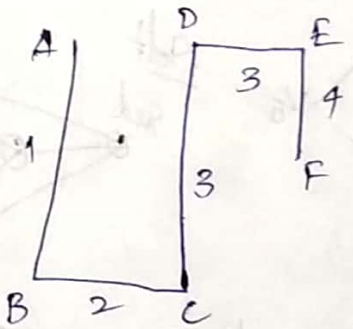
i) Kruskal's algorithm

ii) Prim's algorithm.

*



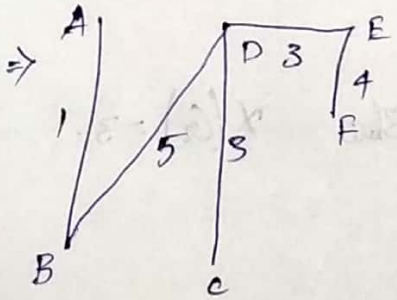
⇒



S1

weight = 13

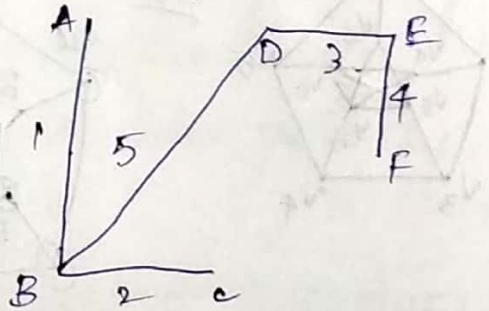
⇒



weight = 16

S2

⇒



weight = 15.

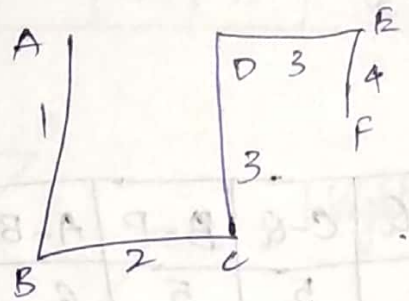
S3

∴ MST = S1.

⇒ from the given graph, we can construct 3 spanning trees (S1, S2, S3). Their weights are 13, 16, 15.

⇒ Out of 3 spanning trees, S1 is minimal spanning tree because it's weight is 13.

∴ Required MST is



1. KRUSKAL'S ALGORITHM :-

→ It is one of the algorithms to find MST.

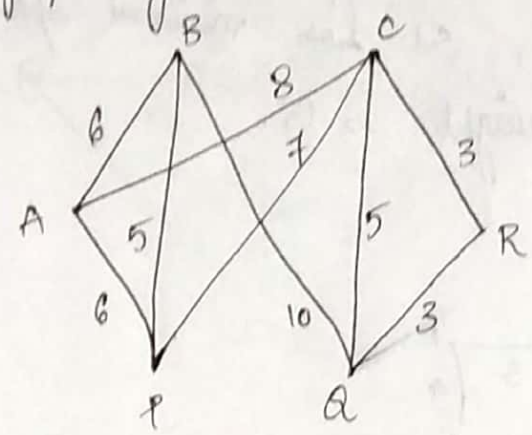
Step-1: Given a connected, weighted graph 'G' with 'n' vertices, list the edges of 'G' according to their weights in ascending order.

Step-2: Starting with smallest edge, proceed sequentially one edge at a time such that no cycle is produced.

Step-3: Repeat step-2 until all vertices are present in spanning tree.

NOTE: Kruskal's algorithm is also called as greedy process.

* Find a MST using Kruskal's algorithm of weighted graph given below.



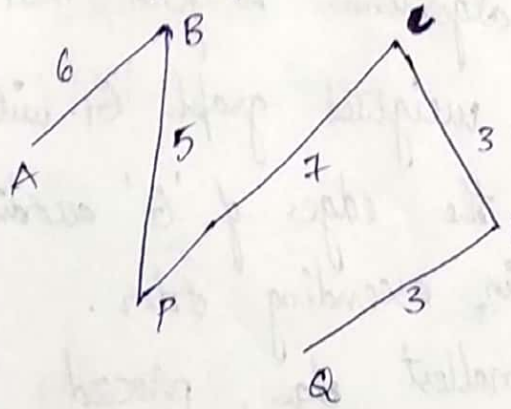
Sol:

Edge	C-R	R-Q	C-Q	B-P	A-B	A-P	C-P	A-C	B-
Weight	3	3	5	5	6	6	7	8	10

No. of edges need to be removed are $|E|-|V|-1$

$\Rightarrow 9-6-1$

$\Rightarrow 4.$

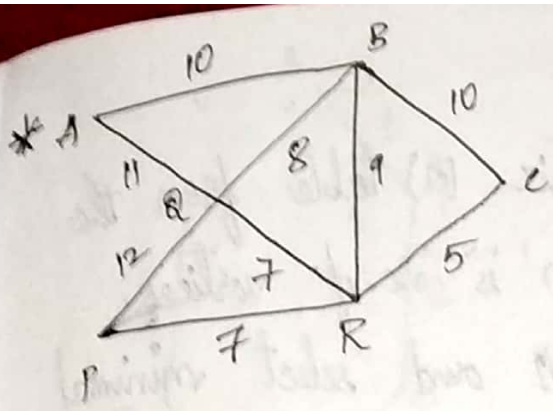


weight of MST = 24.

i) Construct edge C-R with weight '3', from 'R' Construct R-Q with '3' units.

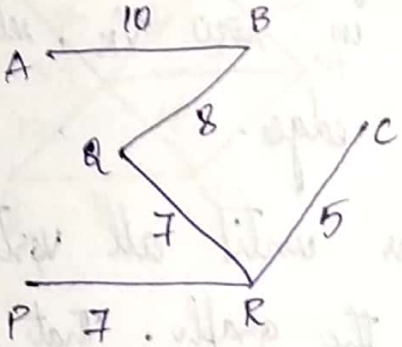
ii) Consider 'C' as root, Construct edge from C-P with '7' units.

iii) Construct the edges P-B & B-A with 5,6 weights respectively.

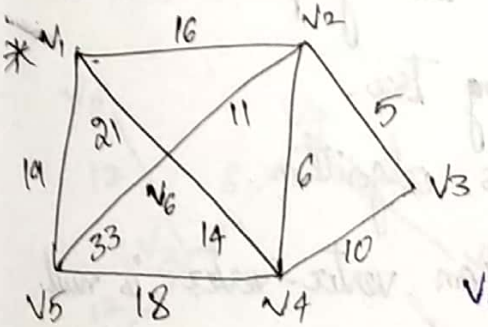


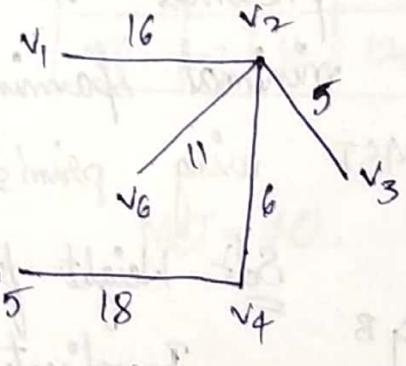
Sol

Edge	C-R	P-R	Q-R	B-Q	B-R	A-B	B-C	A-Q	P-Q
Weight	5	7	7	8	9	10	10	11	12



Weight of MST = 37.





MST = 56.

Sol

Edge	V2-V3	V2-V4	V3-V4	V2-V6	V4-V6	V1-V2	V4-V5
Weight	5	6	10	11	14	16	18
Yes/No	✓	✓	×	✓		✓	✓

V1-V5	V1-V6	V5-V6
19	21	33
×	×	×

2. PRIM'S ALGORITHM:-

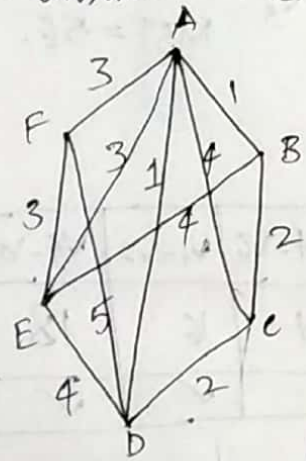
Step-1: Construct n x n matrix (or) table from the given graph where 'n' is no. of vertices.

Step-2: Choose a vertex V_1 and select minimal edge in the table in V- row. Draw the edge from $V_1 - V_k$.

Step-3: Repeat the procedure in row V_k, select the minimum weighted edge.

Step-4: Continue this process until all vertices are presented in the graph. That is the minimal spanning tree.

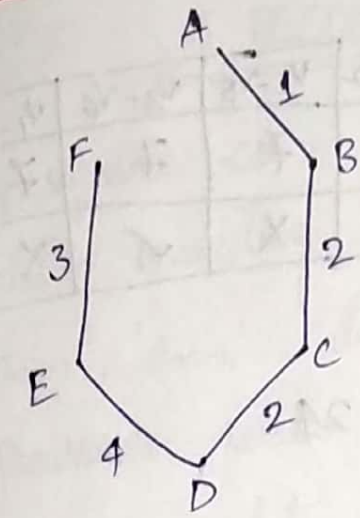
* Construct MST using prim's algorithm.



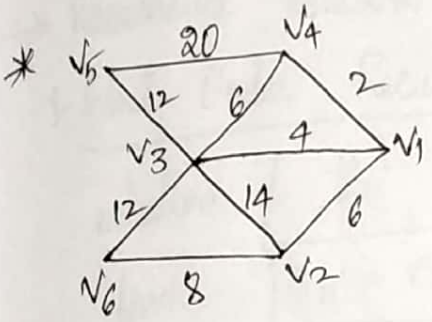
Sol: Weight from vertex-vertex is null.
In undirected graph, it is infinity.

	A	B	C	D	E	F
A	—	1	4	1	3	3
B	1	—	2	∞	4	∞
C	4	2	—	2	∞	∞
D	1	∞	2	—	4	5
E	3	4	∞	4	—	3
F	3	∞	∞	5	3	—

→ If there is no direct edge, it is infinity.

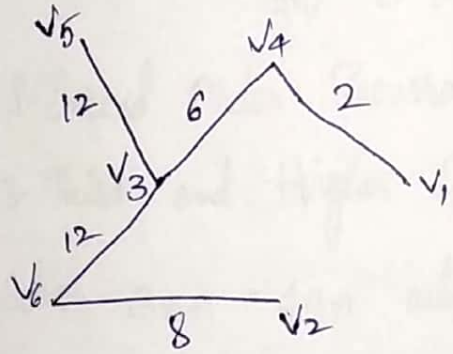


MST = 12.

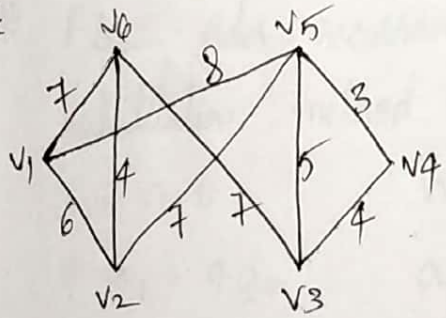
* 

Sol:

	V1	V2	V3	V4	V5	V6
V1	–	6	4	2	∞	∞
V2	6	–	14	∞	∞	8
V3	4	14	–	6	12	12
V4	2	∞	6	–	20	∞
V5	∞	∞	12	20	–	∞
V6	∞	8	12	∞	∞	–

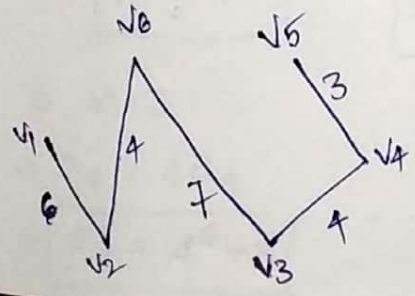


MST = 40.

* 

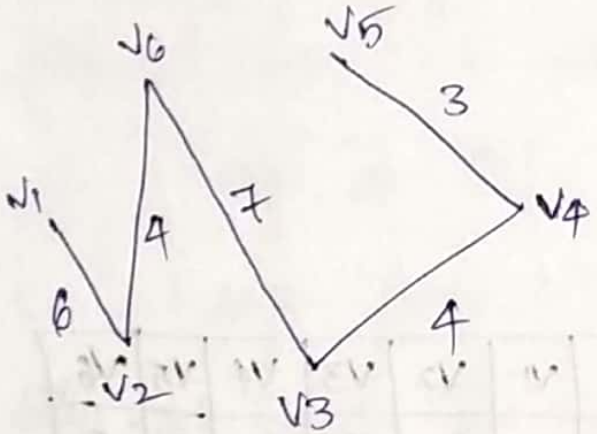
Sol:

	V1	V2	V3	V4	V5	V6
V1	–	6	∞	∞	8	7
V2	6	–	∞	∞	7	4
V3	∞	∞	–	4	5	7
V4	∞	∞	4	–	3	∞
V5	8	7	5	3	–	∞
V6	7	4	7	∞	∞	–



MST = 24.

Edge	v_4-v_5	v_3-v_4	v_2-v_6	v_3-v_5	v_1-v_2	v_2-v_5	v_3-v_6	v_1-v_6	v_1-v_4
Weight	3	4	4	5	6	7	7	7	8
Yes/No	✓	✓	✓	×	✓	×	✓	×	×



MST = 24.