Date :
Page No.

Algorithm of BFS

Step 1 ⇒ Define a queue of size of total no of vertices in the graph.

Step 2 ⇒ Select any vertex as starting point for traversal visit that vertex and insert it into the queue.

Step 3 ⇒ Visit all the non-visited adjacent vertices of the vertex which is at the front of the queue and insert them into the queue.

Step 4 ⇒ When there is no new vertex to be visited from the vertex which is at the front of the queue then delete that vertex.

Step 5 ⇒ Repeat step 3 & 4 until queue becomes empty.

Step 6 ⇒ When queue becomes empty then produce final spanning tree by removing unused edges from the graph.

Date :
Page No.

Unit - 4

Graphs

A graph is a data structure that consist of the following two components:

1. A finite set of ~~vertices~~ vertices is also called as nodes.
2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not the same as (v, u) in case of a directed graph (di-graph).

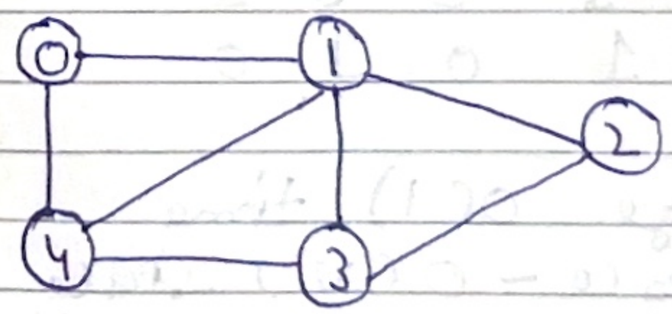
(u, v) → indicates that there is an edge from vertex u to vertex v.
The edge may contain weight/value/cost.

Use of Graphs:

⇒ Represent networks
⇒ Include Path in a city or telephone network.
⇒ Used in social networks like LinkedIn, Facebook.

FB → Person as node
→ Contain info like Person id, name, gender etc

Ex:



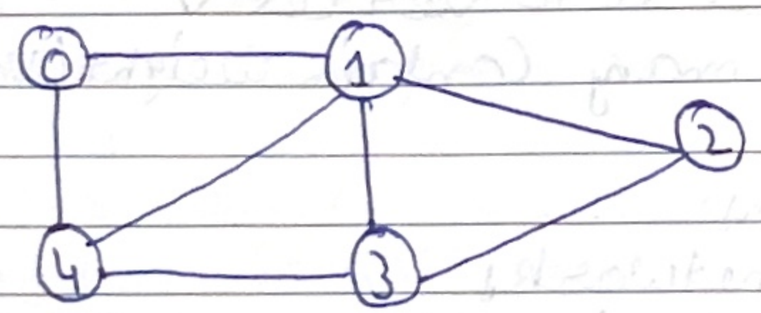
Date : / /
Page No.

Representation of graph:

⇒ Adjacency Matrix
⇒ Adjacency list

☆ Adjacency Matrix: Adjacency Matrix is a 2D array of size V x V where V is the no of Vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j]=1 indicates that there is an edge from Vertex i to Vertex j.

- Adjacency matrix for Undirected graph is always symmetric.
- If adj[i][j] = w, then there is an edge from Vertex i to Vertex j with weight w.

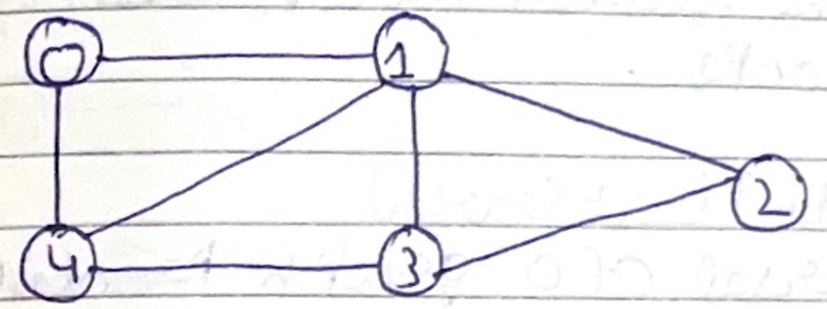


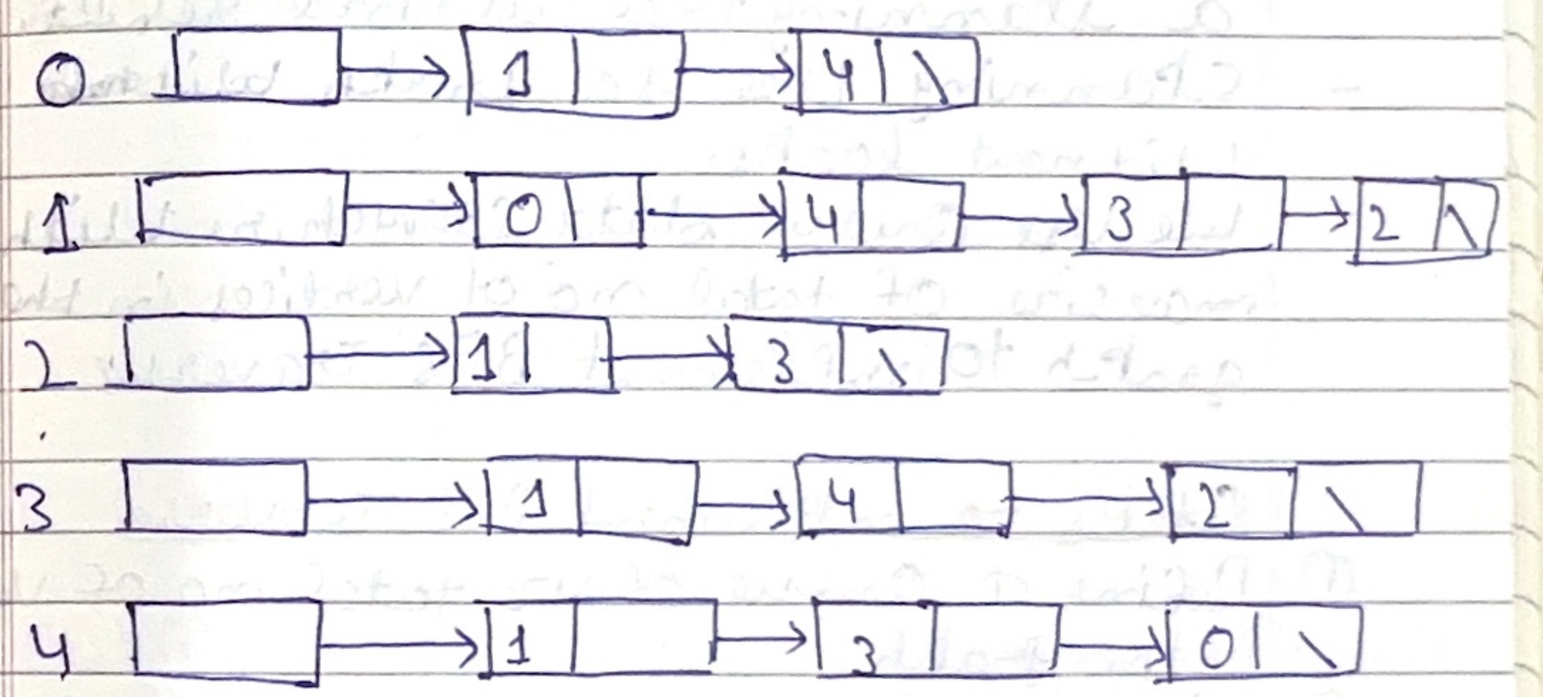
	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

Removing Edge O(1) time
Space - O(V²) space
Adding a vertex - O(V²) time

Date : / /
Page No.

☆ Adjacency List: An array of list is used The size of array is equal to the no of Vertices. This representation can also be used to represent a weighted graph.





Space - O(|V| + |E|)
Worst case - O(V²)

Query like where there is an Edge O(V) time

Graph Traversal

BFS DFS

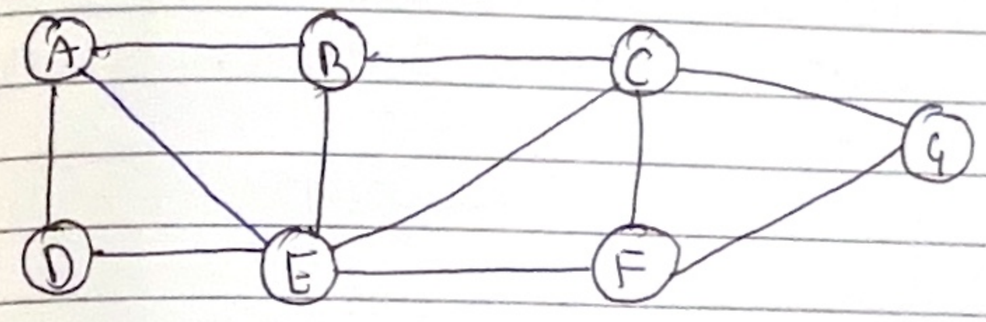
Graph Traversal is a technique used for searching a vertex in a graph.

- Graph Traversal is also used to decide the order of vertices is visited in the search process.
- Graph Traversal find the edges to be used in the search process without creating loops.

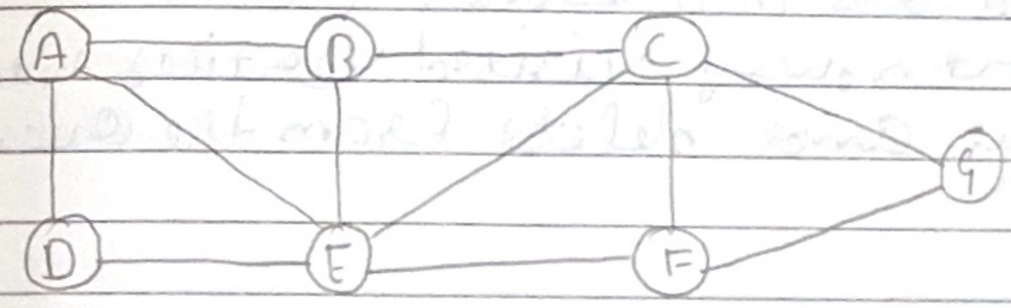
⇒ BFS (Breath First Search)
- BFS traversal of a graph produces a spanning tree as final result.
- Spanning tree is a graph without loops.
- We use Queue data structure with max size of total no of vertices in the graph to implement BFS Traversal.

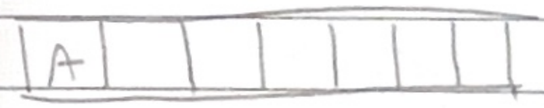
Steps to implement BFS Traversal
1. Define a Queue of size total no of vertices in the graph.
2. Select any vertex as starting point for traversal. Visit that vertex and insert it into the Queue.
3. Visit all the non-visited adjacent vertices of the vertex which is at front of the Queue and insert them into the Queue.
4. When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.
5. Repeat steps 3 and 4 until queue becomes empty.
6. When queue becomes empty, then produce final spanning tree by removing unused edges from the graph.

Q1

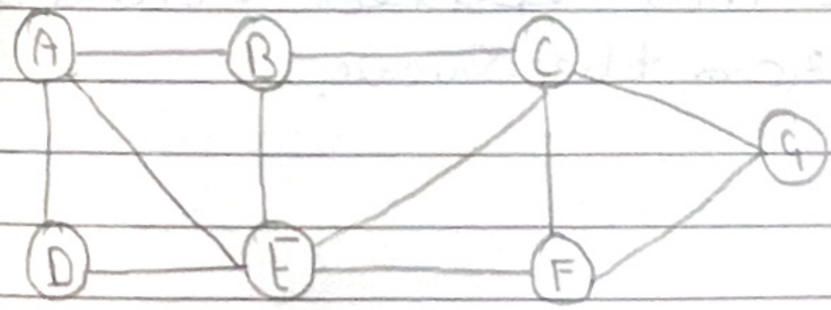


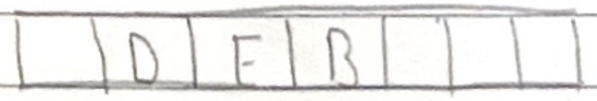
Sol Step 1 Select the vertex A as start point (visit A) Insert A into the Queue.





Step 2 Visit all adjacent vertices of A which are not visited (D, E, B)
Insert newly visited vertices into the Queue & delete A from the queue

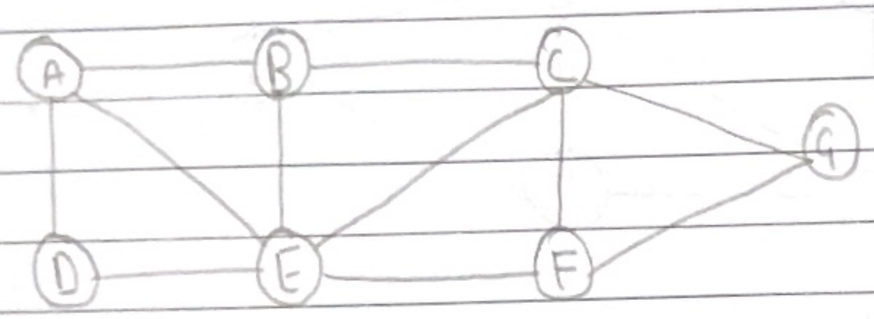




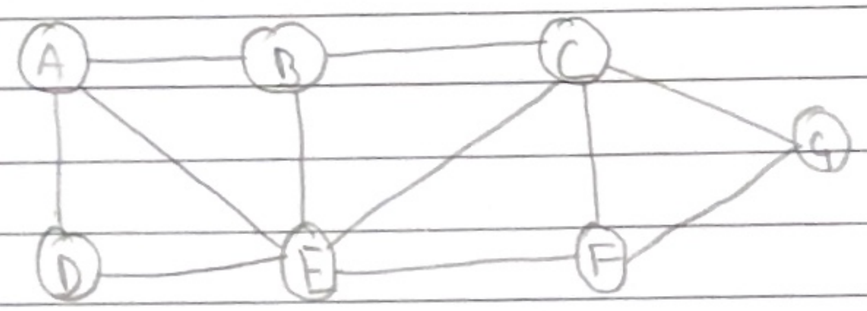
Step 3 Visit all adjacent vertices of D (which are not visited (there is no vertex)

- Delete D from the Queue

			E	B			

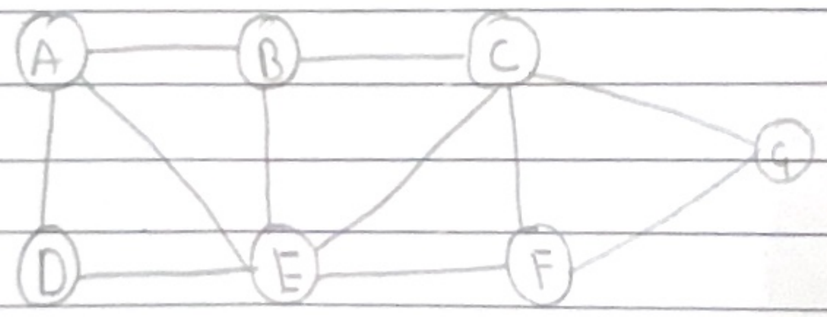


Step 4 Visit all adjacent vertices of E which are not visited (C, F)
- Insert newly visited vertices into the Queue and delete from the Queue.



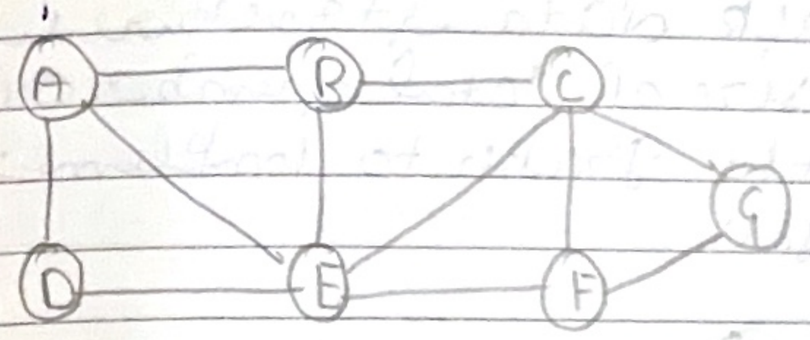
			B	C	F		

Step 5 Visit all adjacent vertices of B which are not visited (there is no vertex)
- Delete B from the Queue.



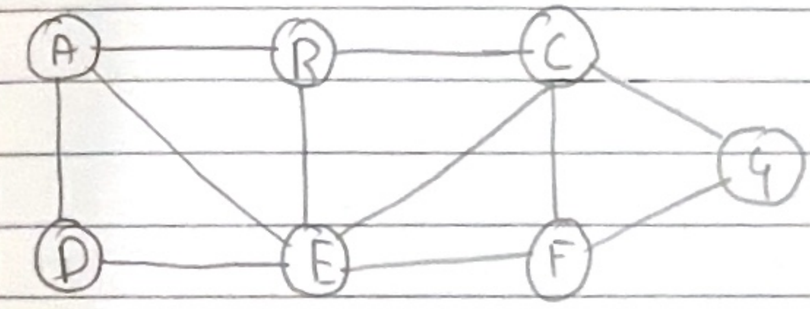
				C	F		

Step 6 Visit all adjacent vertices of C which are not visited (G)
- Insert newly visited vertex into the Queue and delete C from the Queue.



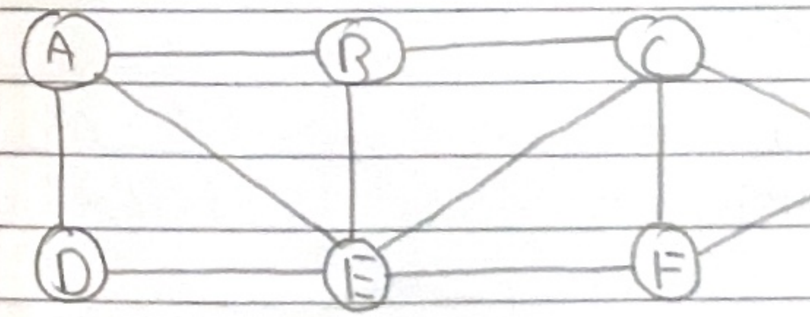
						F	G

Step 7 Visit all adjacent vertices of F which are not visited (there is no vertex)
- Delete F from the Queue.

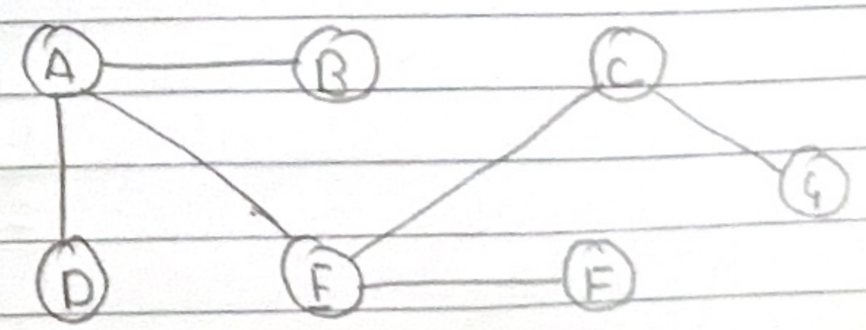


							G

Step 8 Visit all adjacent vertices of G which are not visited (there is no vertex)
- Delete G from the Queue.



Result



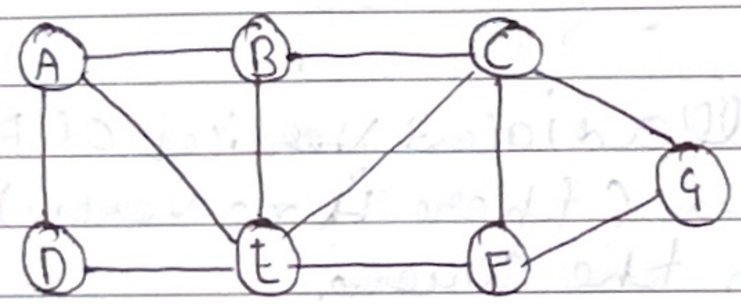
Date : / /
Page No.

DFS (Depth First Search)

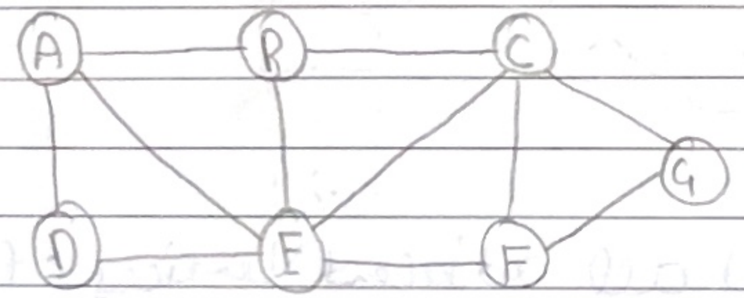
DFS traversal of a graph produces a spanning tree as final result.

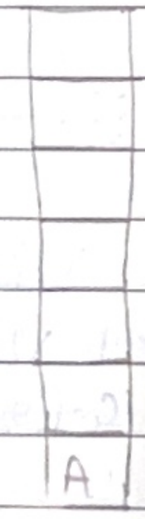
- Spanning tree is a graph without loops.
- We use stack data structure with maximum size of total number of vertices in the graph to implement DFS traversal.

(1)

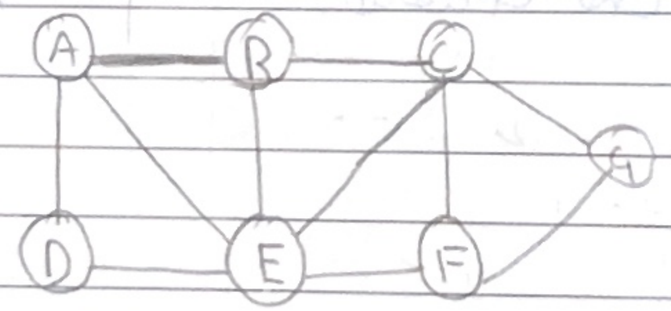


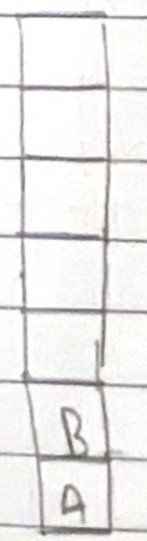
Step (1)





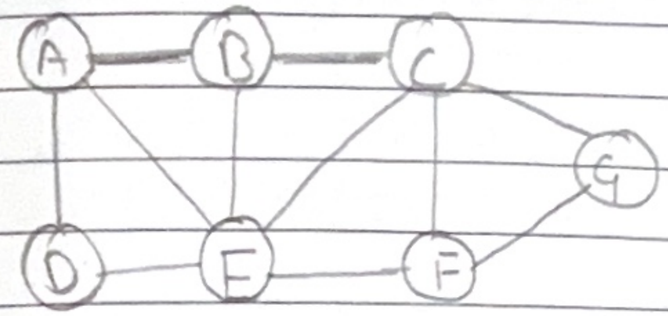
Step-2

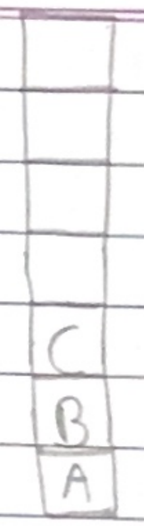




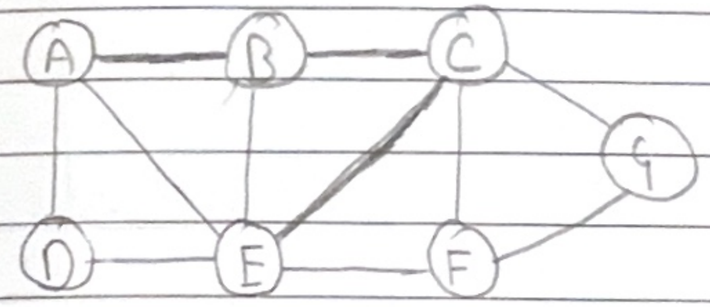
Date : / /
Page No.

Step-3

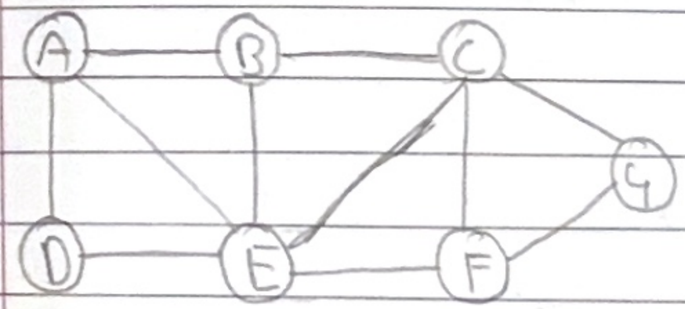




Step-4

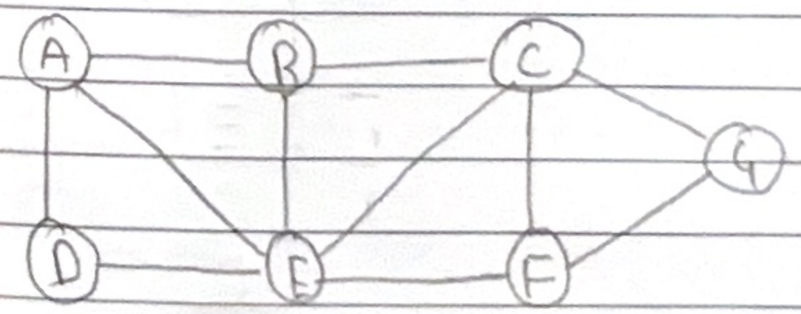


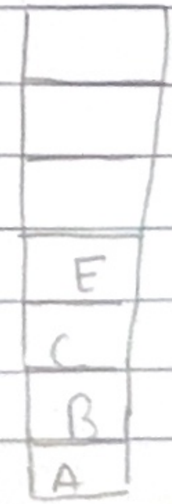
Step-5



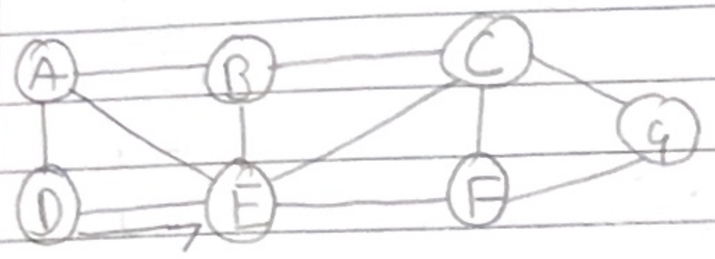
Step-6 There is no new vertex to be visited from D. So use backtrack

- Pop D from the stack



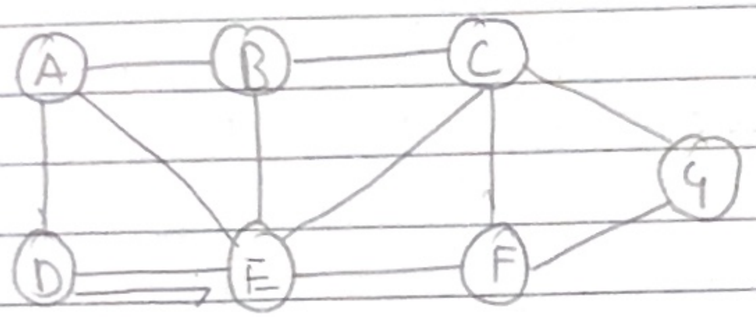


Step-7 Visit adjacent vertex of F which is not visited (E).
- Push F onto the stack



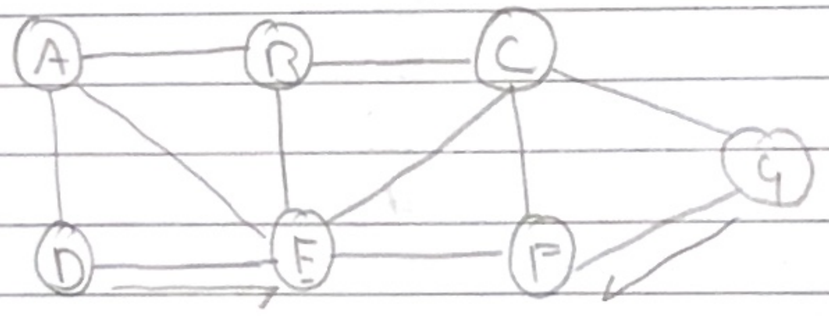
F
E
C
B
A

Step-8



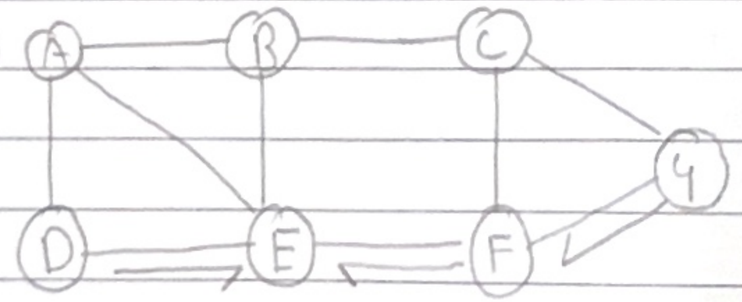
G
F
E
C
B
A

Step-9 There is no new vertex to be visited from G, so use backtrack
- Pop G from the stack



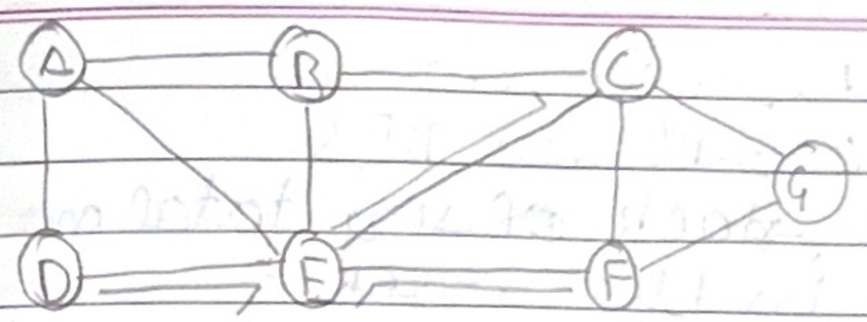
F
E
C
B
A

Step-10



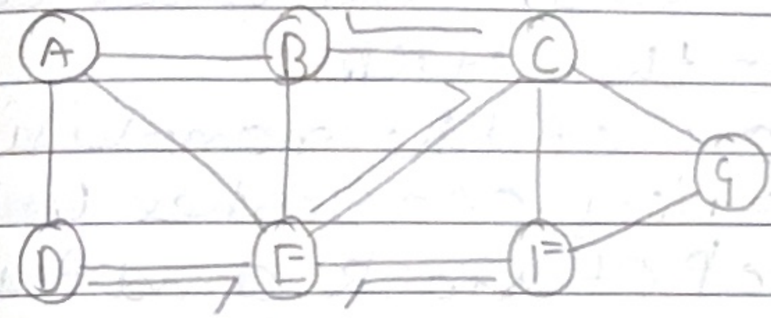
E
C
B
A

Step-11



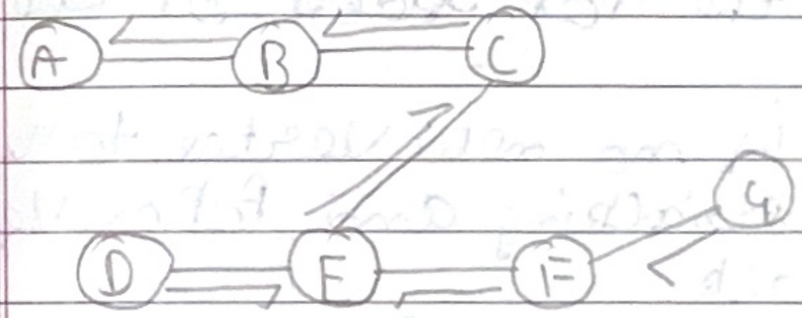
C
B
A

Step-12



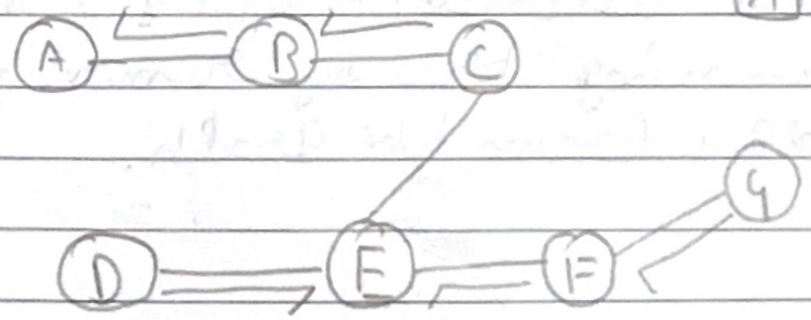
B
A

Step-13

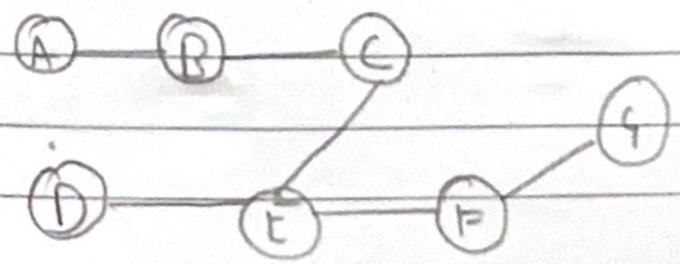


A

Step-14



Stack becomes empty. So stop DFS traversal. &
Final MST Of DFS



Date :
Page No.

Steps to implement DFS

Step ① Define a stack of size total no of vertices in the graph.

Step ② Select any vertex as starting point for traversal visit that vertex and Push it onto the stack.

Step ③ Visit anyone of the non-visited adjacent vertices of a vertex which is at the top of stack and Push it onto stack.

Step ④ Repeat step ③ Until there is no new vertex to be visited from the vertex which is at the top ~~stack~~ of the stack.

Step ⑤ When there is no new vertex to visit then use backtracking and Pop one vertex from the stack.

Step ⑥ Repeat step 3, 4 and 5 Until stack becomes empty.

Step ⑦ When stack becomes empty, then Produce final spanning tree by removing unused edges from the graph.

Date :
Page No.

Graph Theory

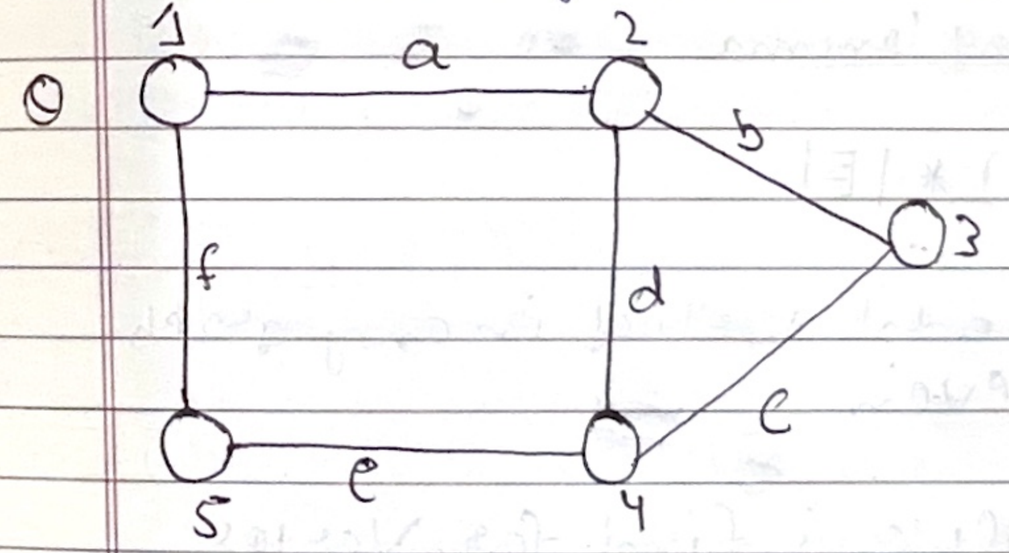
$G = (V, E)$

V = Vertex Set $\{V_1, V_2, V_3, V_4, \ldots V_n\}$
E = Edge Set $\{E_1, E_2, E_3 \ldots E_n\}$

in which $l_k \in E$ is $l_k = \{V_i, V_j\}$ is called Undirected graph.

$V_i \xrightarrow{E_1} V_j$ is called directed graph
$V_i \overset{E_1}{—} V_j$ it's called Undirected graph

$|V|$ = order of graph (no of vertex)
$|E|$ = Size of graph (no of Edge's)



$G = (V, E)$
$V = \{1, 2, 3, 4, 5\}$
$E = \{a, b, c, d, e, f\}$

Adjacent Vertex = Common edge b/w two vertex

$a = \{1, 2\}$

Adjacent Edge: Used in two commonly vertex
a is in common edge

Date :
Page No.

Self loop - Edge have start and End point are same.

Multi edge - Multiple Edges b/w two Vertices

Pseudograph - Graph Contain self loop & multi edge

Multigraph - Graph have multiple edges b/w two Vertices known as multigraph

Simple Graph - Not Contain self loop, multi Edges.

* Handshaking lemma

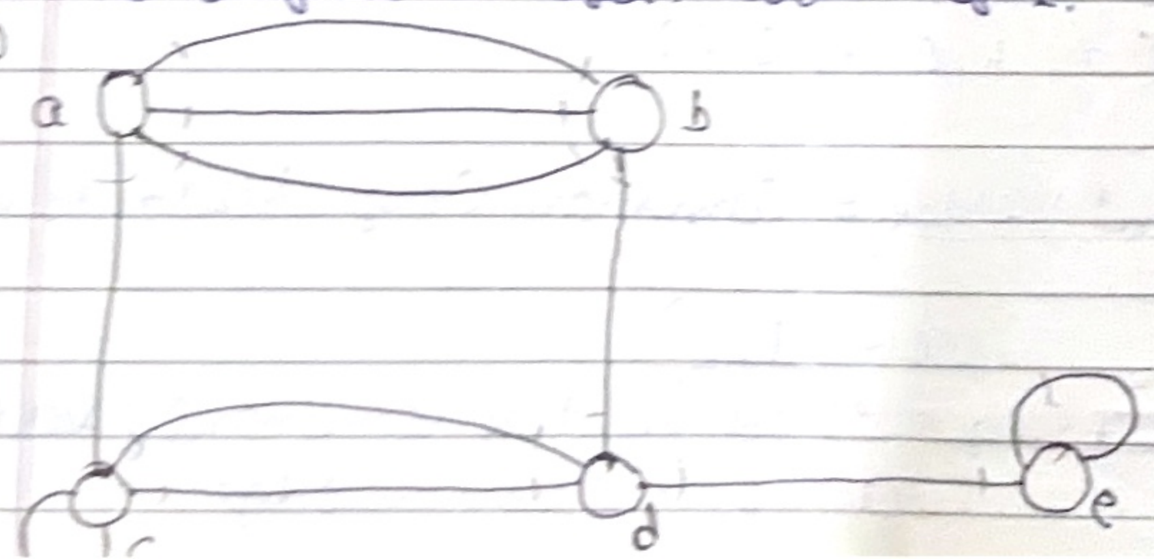
$\sum d(v) = 2 * |E|$

The no of odd Vertices in any graph it always even

Degree is always find for Vertex

The degree of self loop is 2.

Q



Date :
Page No.

Sol Degree of vertex a = 4
Degree of vertex b = 4
Degree of vertex c = 5
Degree of vertex d = 4
Degree of vertex e = 3

Sum of degree of Vertex = 4+4+5+4+3
= 20

Total no of edge = 10

Self loop (a, a) one degree

$\sum d(V) = 2 * |E|$
$20 = 2 \times 10$
$20 = 20$

* Havel-Hakimi Theorem

For a given degree sequence, whether a simple graph exist or not.

Q (2, 2, 2, 2)

Arrange in Descending order

(2, 2, 2, 2) Remove First element and subtract -1 from the elements remove

(1, 1, 2)

(2, 1, 1)

(1, 1)

(0, 0) graph exist

Date : / /
Page No.

⊙ (2, 3, 0, 1, 1)

Sol Arrange in descending order

(2, 2, 1, 1, 0)
  -1

(1, 0, 0, 0)
  -1

(-1, 0, 0) Graph not exist

⊙ (6, 5, 4, 3, 3, 1)

(5, 4, 3, 3, 1)
  -1

Not exist graph because we delete 6
Subtract -1 in 6 elements but element is 5

⊙ (7, 6, 5, 4, 4, 3, 2, 1)

(6, 5, 4, 4, 3, 2, 1)
  -1

(5, 4, 3, 3, 2, 1, 0)
  -1

(3, 2, 2, 1, 0, 0)
  -1

(1, 1, 0, 0, 0)
  -1

(0, 0, 0, 0) Graph exist

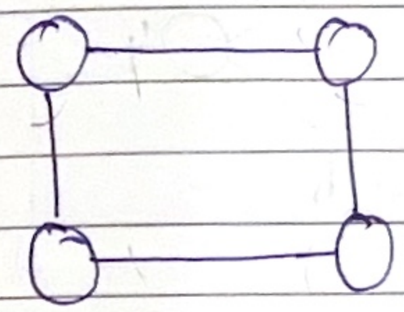
Date : / /
Page No.

Regular Graph

A Vertex has same degree in graph is called regular graph.

No of Edges $E = \frac{n \times d}{2}$

n = no of Vertex
d = degree of each Vertex

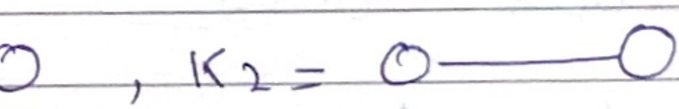


$E = \frac{4 \times 2}{2} = \frac{8}{2} = 4$

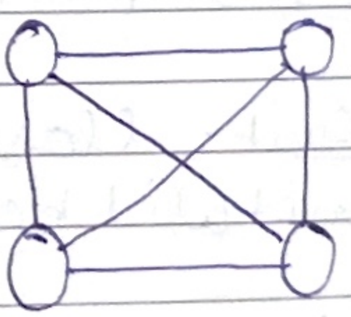
Complete graph

A simple graph is Complete graph (K_n) in which every pair of vertices are adjacent.

K_n = where n is no of vertices

K_1 =  , K_2 = 

K_3 = 

K_4 = 

No of Edges = $^nC_2 = \frac{n \times (n-1)}{2} = 6$

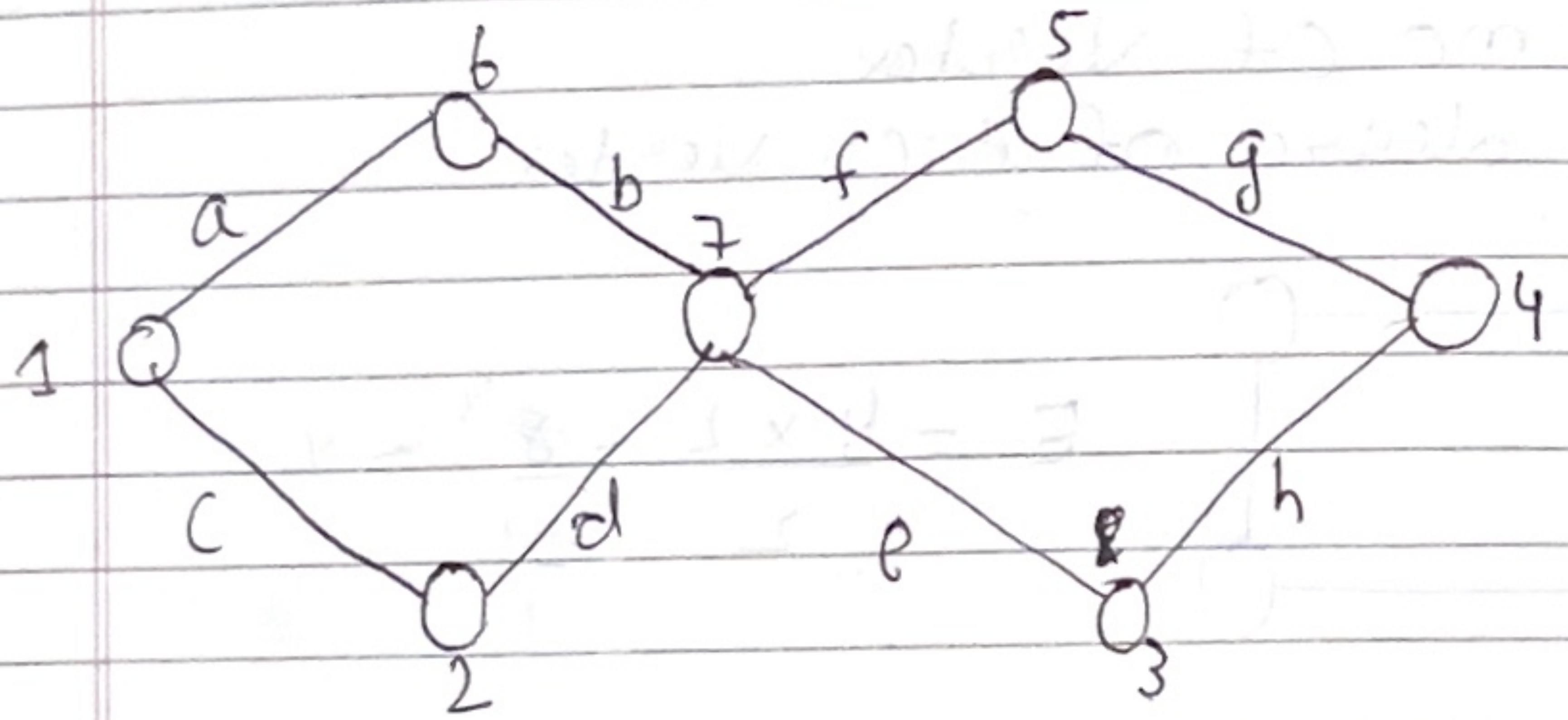
$^5C_2 = \frac{5 \times (4)}{2} = 10$

Degree of Every vertex = n-1

The no of vertex are 'n' how many simple graph Possible = $2^E = 2^{\frac{n(n-1)}{2}}$

* Every Regular graph is not Complete graph

* Every Complete graph is regular graph.



Walk - Repetion are allowed. opened walk and closed walk. open means start kaha se bhi end; Close mean Jaha se start bhi wa end
1a6b7d

Trail - Edges repetition not allowed
Vertex repetion allowed.

7 5 4 3 7 ,

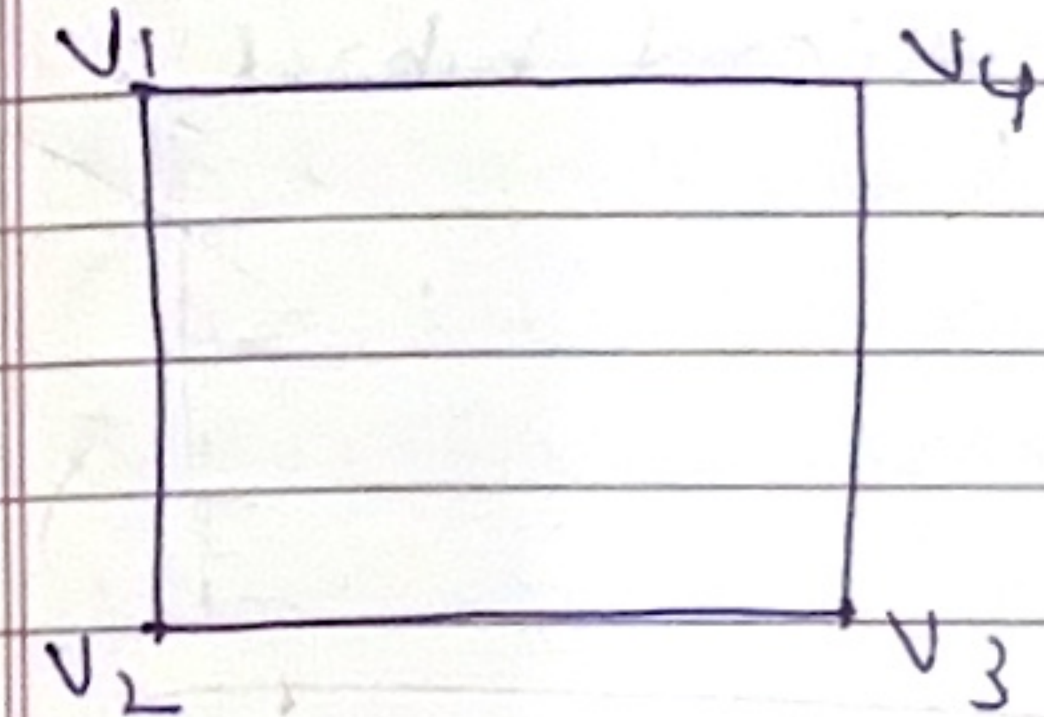
Circuit - Closed Trail also known as circuit without repeat edge

1 6 7 5 4 3 7 2 1

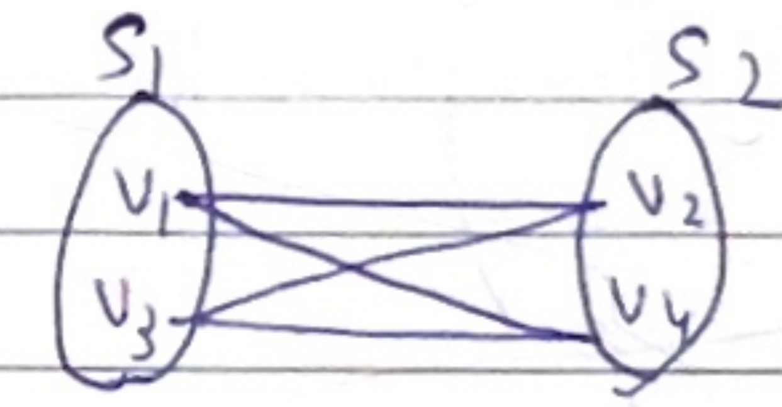
Path - vertex repetion not allowed
1 6 7 2 1
Vertices start and end vertices are same
closed Path - Cycle

Date : / /
Page No.

Bipartite Graph
'G' is bipartite if its vertex set 'V' can be Partitioned into two disjoint non empty set V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2 so that no edge in G connects either two vertices in V_1 or two vertices in V_2

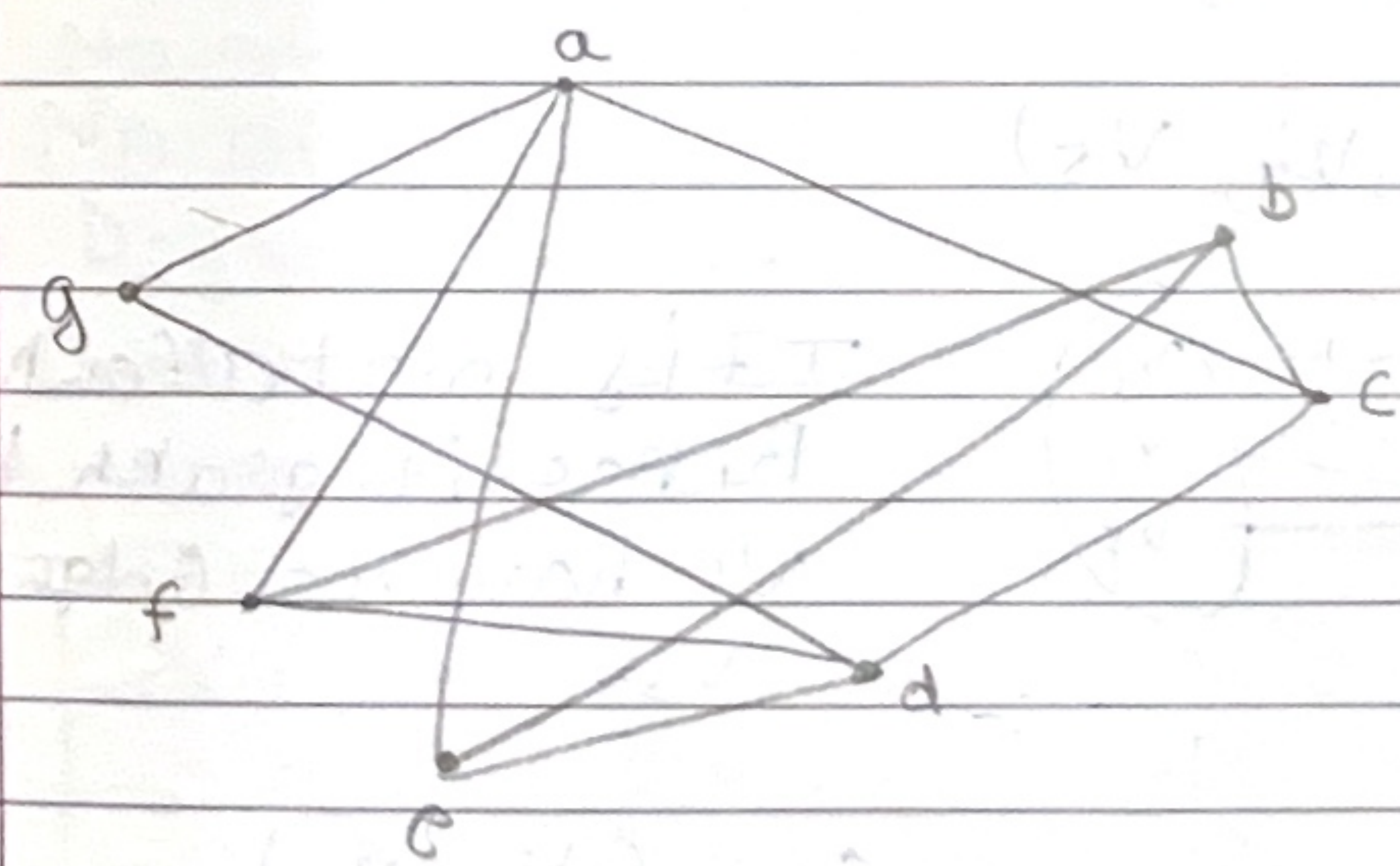


$S_1 = \{V_1, V_3\}$ $S_2 = \{V_2, V_4\}$

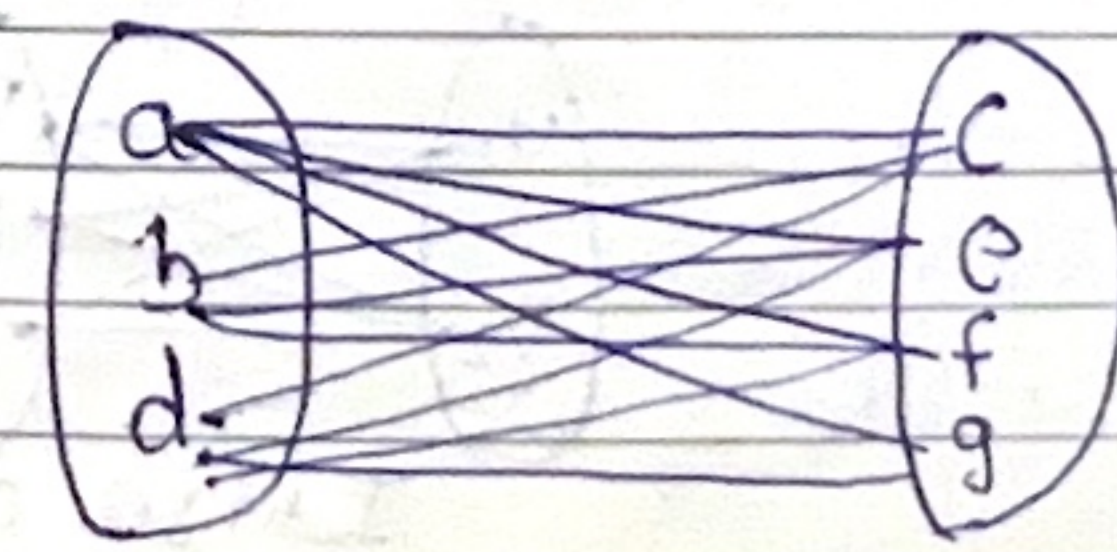


* Take the set in which not common edge in b/w two vertex in same set

Q



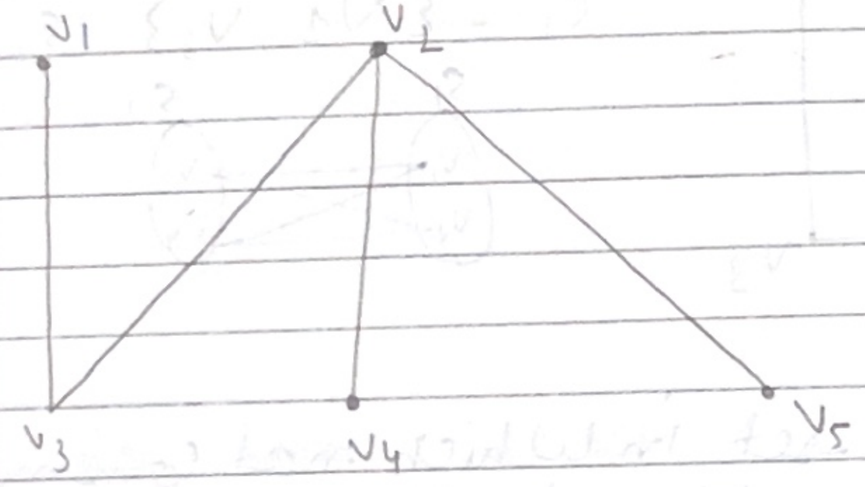
V(G) = (a, b, c, d, e, f, g)



Date :
Page No.

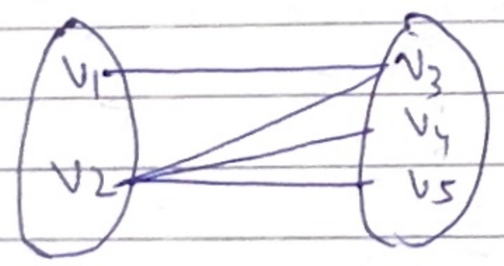
Complete Bipartite Graph: If it has its Vertex set Partitioned into two subsets of 'm' and 'n' Vertices respectively. There is an edge b/w two vertices if and only if one vertex is in first subset and other in second subset.

* It means Vertex of first subset is Completely connected Vertices of second subset.

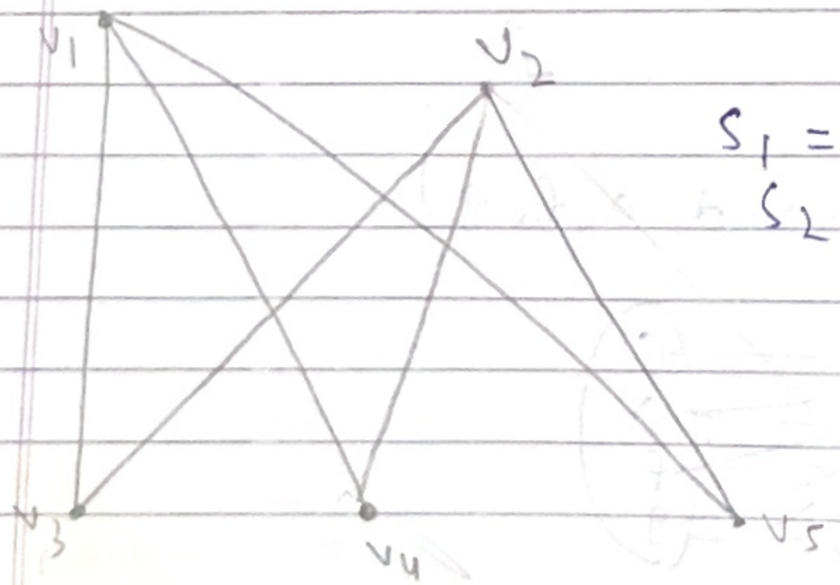


$S_1 = (V_1, V_2)$

$S_2 = (V_3, V_4, V_5)$

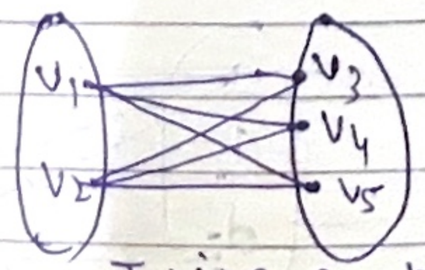


It is not a Complete Bipartite graph because V_1 have no edge b/w V_5



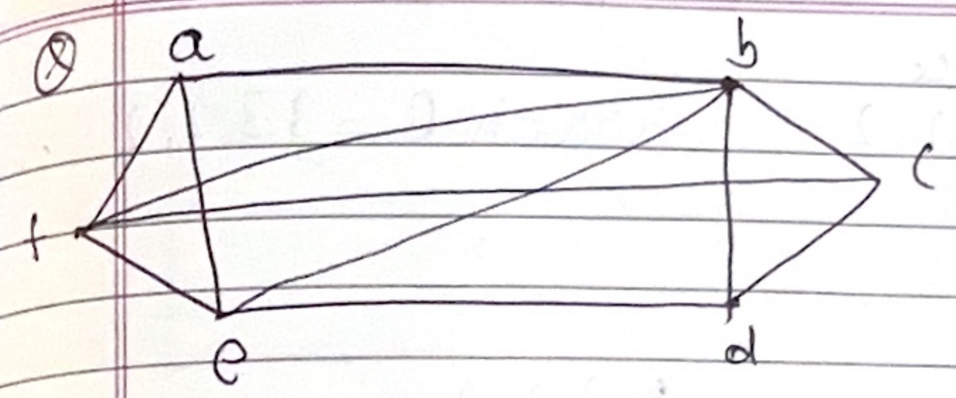
$S_1 = (V_1, V_2)$

$S_2 = (V_3, V_4, V_5)$



It is a Complete bipartite

Date :
Page No.



$S_1 = (a, d, c)$

but d & c have common edge

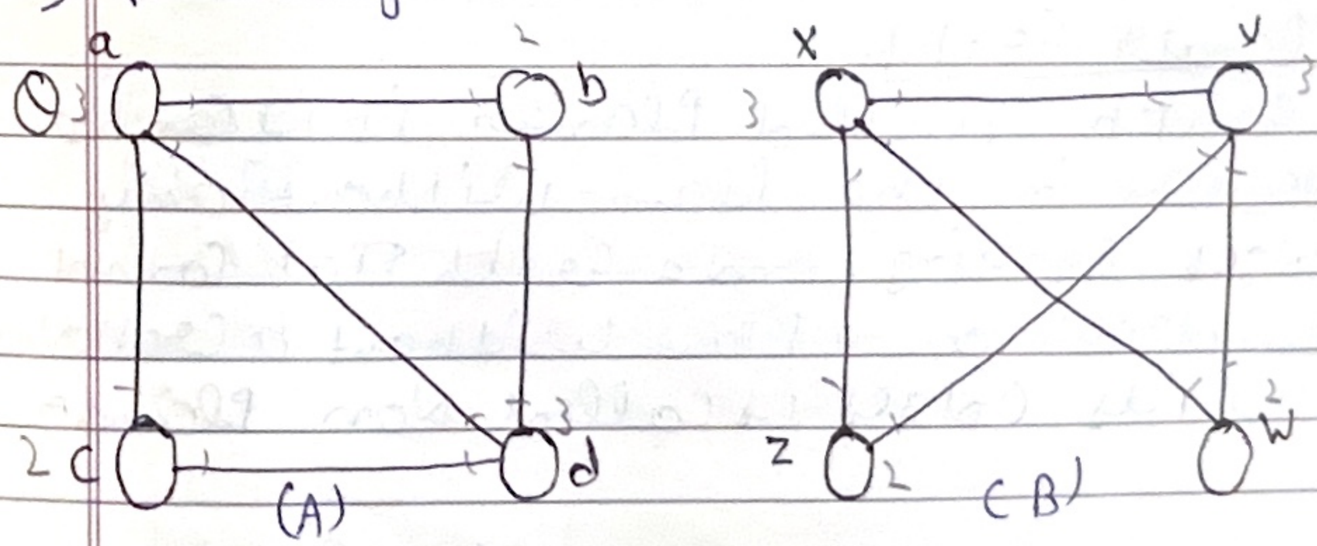
Not a Bipartite graph: it's Vertex Can't be Partitioned in two sets such that edges Can't Connect two vertices from same subset

* **Isomorphic Graph**

Two graph are different to each other but they are same known as isomorphic

Check the properties the graph is isomorphic:

⇒ No of vertex
⇒ No of Edge
⇒ Degree sequence
⇒ Mapping



Sol No of Vertex Graph A = 4, Graph B = 4
No of Edge Graph A = 5, Graph B = 5

Date : / /
Page No.

Degree Sequence
Graph A = 3,3,2,2 Graph B = 3,3,2,2

Matching

a has degree sequence 3, 2, 2
b has degree sequence 3, 3
c has degree 2 Connected sequence 3, 3
d has degree 3 Connected sequence 3, 2, 2

x has degree 3 Connected sequence 3, 2, 2
y has degree 3 Connected sequence 3, 2, 2
w has degree 2 Connected sequence 3, 3
z has degree 2 Connected sequence 3, 3

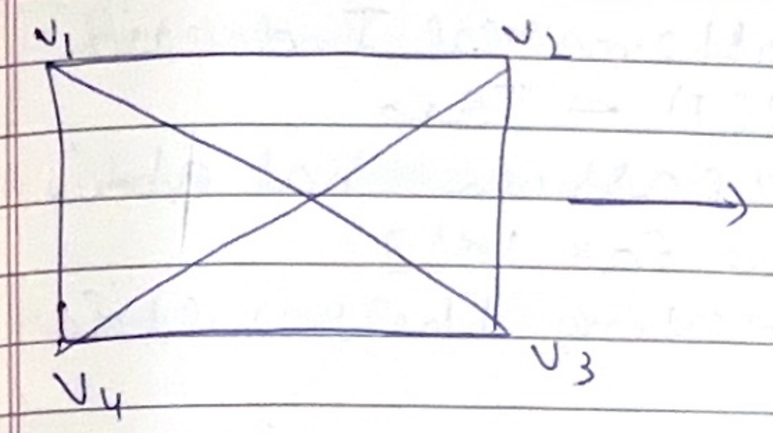
Edge wise a & b: a = x, x & z have edge; b = z, z & w have not edge
b & c have no edge: c = w, w & y have edge
c & d have edge: d = y

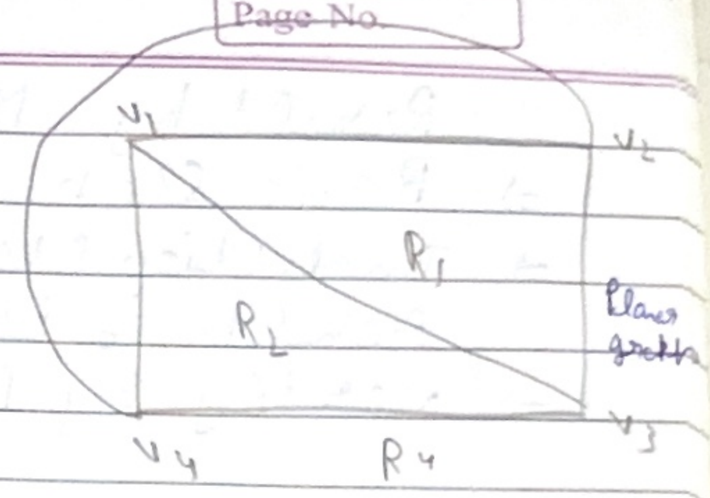
Thus the graph is isomorphic

Planar Graph
A graph is called Planar if it can be drawn in the plane without any edges crossing. And a graph that cannot be drawn on a plane without a crossover b/w its edges is called Non Planar

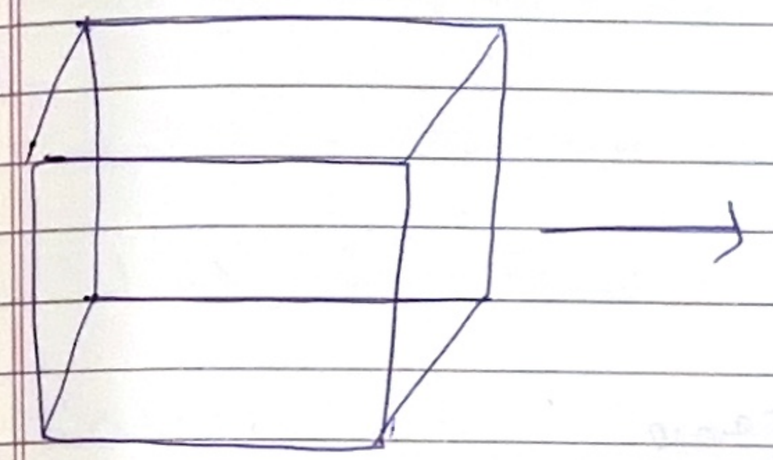
Date : / /
Page No.

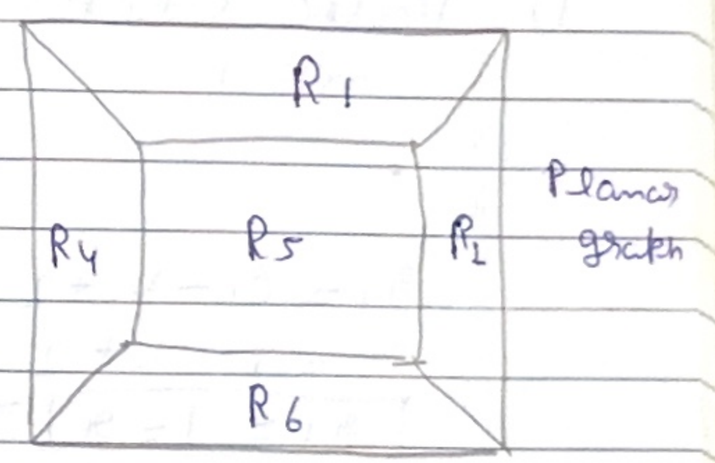
i)





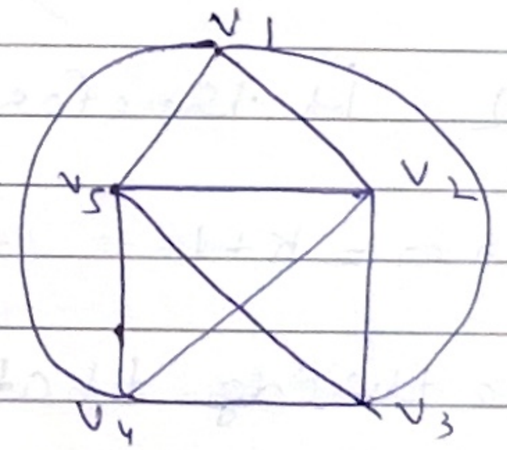
ii)





R6 → outer regia

Theorem: A Complete graph of five Vertices is Non-Planar.



Non Planar graph

VVImp
Euler's formula Proof
Let G be a Connected Planar Simple Graph with 'e' edges and 'v' vertices. Let r be no of regions in a Planar representation of G.

$$r = e - v + 2$$

Proof:

Result by Mathematical Induction.

⇒ Basic Step P(1) → True

⇒ Induction Step [Assume that above result is true for 'k']

⇒ verify by Proving the result for k+1

i) Basic Step:

n = 1

•——————•

$r_1 = e_1 - v_1 + 2$

$r_1 = 1 - 2 + 2$

$\boxed{r_1 = 1 = r}$ True

ii) Induction Step:

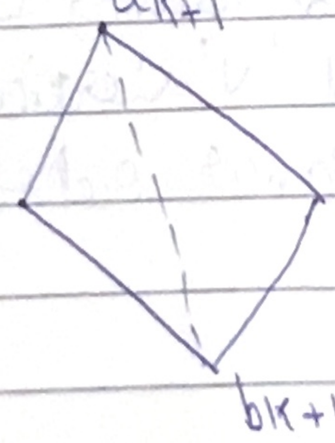
Assume the equation is true for n = k

$r_k = e_k - v_k + 2$ is true for G_k

iii) Verification for n = k+1

let (a_{k+1}, b_{k+1}) be the edge that is added to G_k

Case 1:



Both the vertices are in G_k

$r_{k+1} = r_k + 1$

$e_{k+1} = e_k + 1$

$v_{k+1} = v_k$ both vertices are in G_k

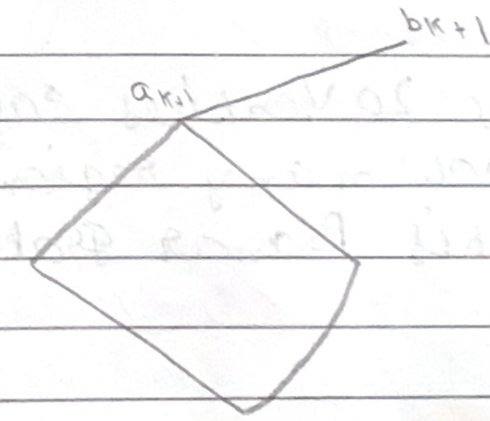
$r_{k+1} = e_{k+1} - v_{k+1} + 2$

$r_k + 1 = e_k + 1 - v_k + 2$

$\boxed{r_k = e_k - v_k + 2}$

It is also True

Case 2:



$r_{k+1} = r_k$

$v_{k+1} = v_k + 1$

$e_{k+1} = e_k + 1$

$r_{k+1} = e_k$

$r_k = e$

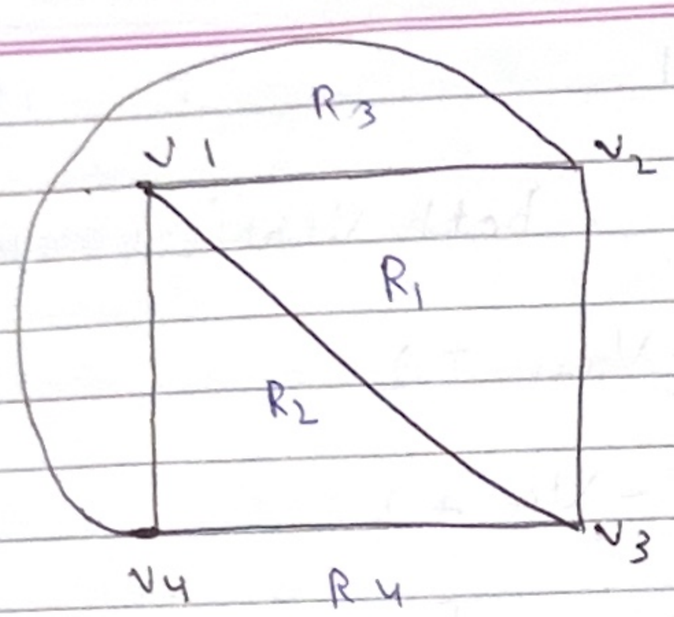
$r_k =$

I

By induc

Date : / /
Page No.

Q



$C = 6 \quad , V = 4$

$r = 6 - 4 + 2$

$\boxed{r = 4}$

Q If there are 20 vertices each of degree 3 then into how many regions does a representation of this Planar graph splits the Plane?

Sol $r = ?$

$V = 20$

$\boxed{\sum deg(V) = 2e}$

degree $= 3$

$\sum deg(V) = 20 \times 3 = 60$

$60 = 2e$

$e = 30$

$r = e - V + 2$

$r = 30 - 20 + 2$

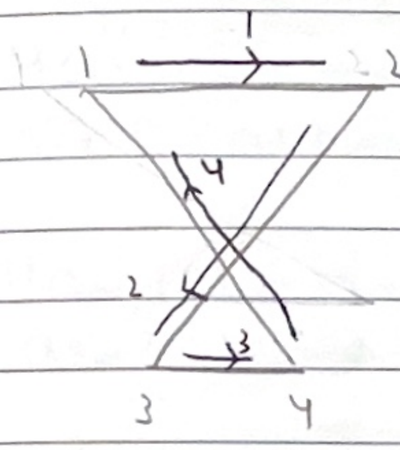
$\boxed{r = 12}$

Date : / /
Page No.

Euler Graph

Euler Path- It is a Path that traverse each edge exactly one and only once. (Vertex can be retraced but Edge can't).

A Graph that Contains an Euler Path is called as Euler Graph.



$1 \to 2 \to 3 \to 4$ it is Euler graph (Each edge visited only once)

$1 \to 2 \to 3 \to 4 \to 1$ It is Euler Circuit

Euler graph is always connected because Euler Path contain all the Edges of the Graph.

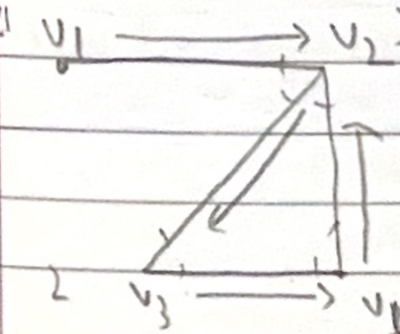
Euler Circuit: First and Last Vertex are same. It is a Circuit that traverse each Edge once & only once.

Imp:-

⇒ A connected Euler graph iff it has atmost 2 odd degree vertices.

⇒ Euler Circuit → Each Vertex is of Even degree.

⇒ Euler Path → Max two vertices odd degree

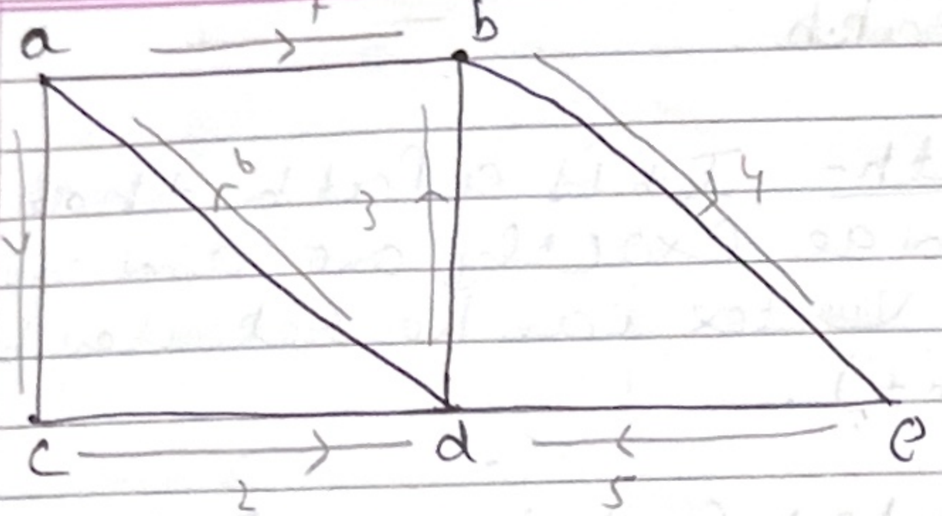


$V_1 \to V_2 \to V_3 \to V_4$ It is a Euler Path.

But not Euler Circuit because Start and End Point are not same.

Date : / /
Page No.

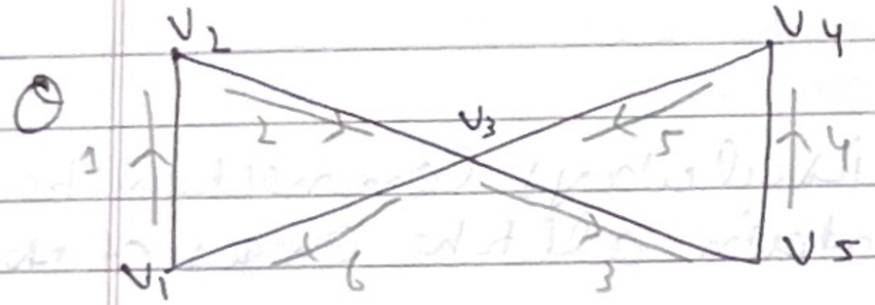
Q



$a \to c \to d \to b \to e \to d \to a \to b$

Degree
a → 3 ⎫
b → 3 ⎭ Two vertex with odd degree
c → 2
d → 4
e → 2

Q

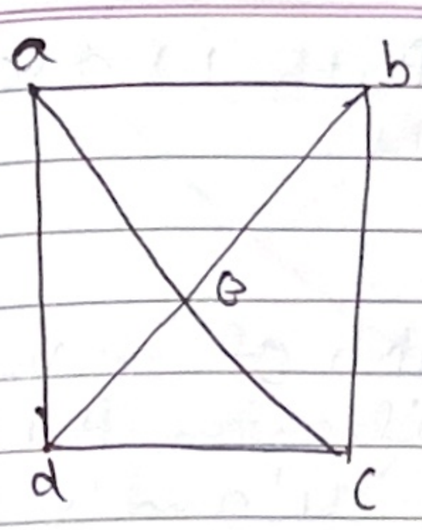


It is an Euler circuit
$v_1 \to v_2 \to v_3 \to v_5 \to v_4 \to v_3 \to v_1$

$v_1 - 2$
$v_2 - 2$
$v_3 - 4$
$v_4 - 2$
$v_5 - 2$
⎭ Even degree

Date : / /
Page No.

Q

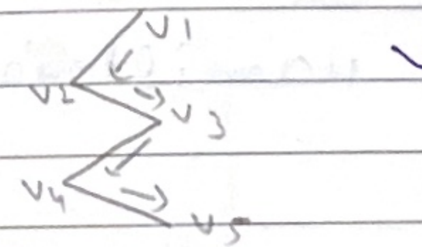


a = 3
b = 3
c = 3
d = 3
e = 4
4 vertices with odd degree
It is not a Euler graph

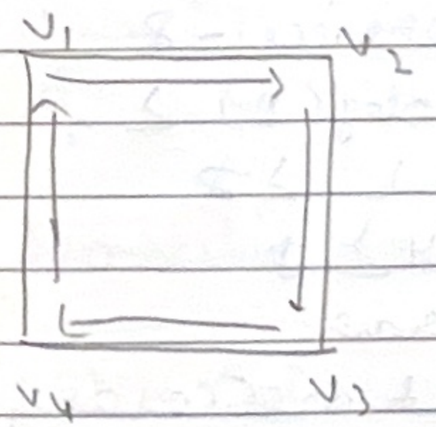
☆ Hamiltonian Path

Hamiltonian path contain each vertex exactly once.

Hamiltonian Circuit: First and last vertex are same



$v_1 \to v_2 \to v_3 \to v_4 \to v_5$
It is an Hamilton Path



$v_1 \to v_2 \to v_3 \to v_4 \to v_1$
It is Hamiltonian Circuit

On removing any one edge from a Hamiltonian circuit, then we are left with Hamiltonian path.

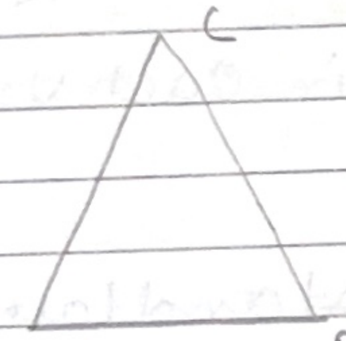
Date :
Page No.

⇒ length of Hamiltonian Path in a connected graph of 'n' vertices is

$\boxed{(n-1) \text{ edges}}$

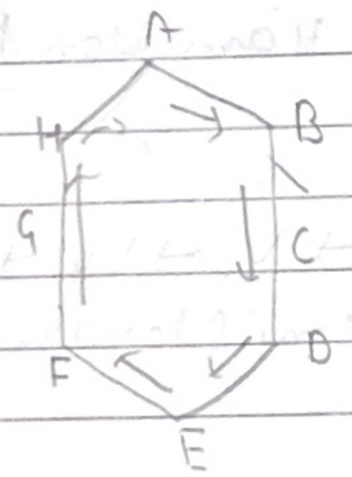
Theorem: Let G be a graph of 'n' vertices then 'G' has a Hamiltonian Path if for only two vertices 'u' and 'v' of G.

$\boxed{\deg(u) + \deg(v) \geq n}$



no of vertices $(n) = 3$
Consider two vertex (A, B)
$\deg(A) = 2$, $\deg(B) = 2$

$\deg(A) + \deg(B) = 4 \geq n$
$4 \geq n$
It is Hamiltonian Path.

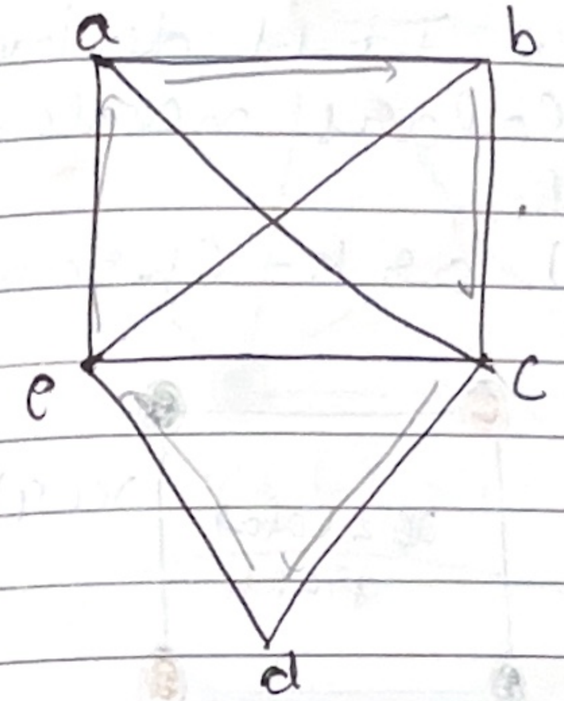


no of vertices $= 8$
$\deg(A) + \deg(D) \geq n$
$3 + 3 \geq 8$
$6 \geq 8$
Not True

This theorem is not necessary true

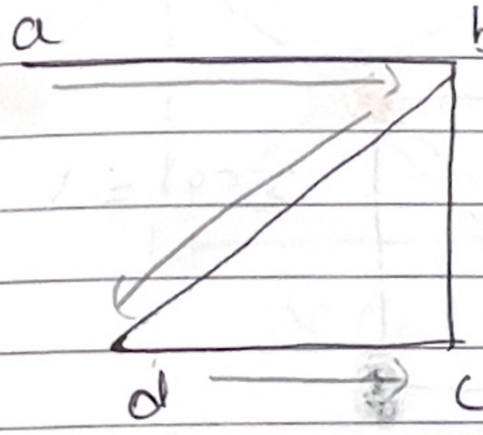
Date :
Page No.

(1)



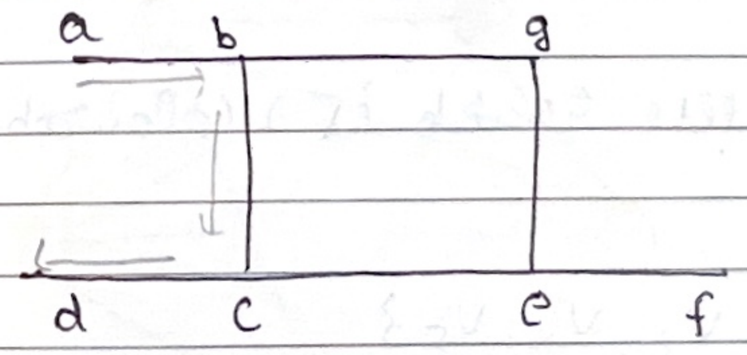
It is an Hamiltonian circuit
[a, b, c, d, e, a]

(2)



It is an Hamiltonian Path
[a, b, d, c]

(3)



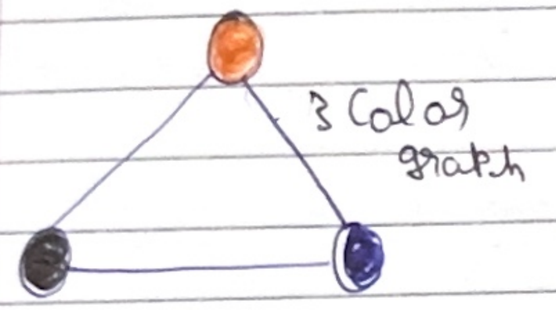
It is an Hamiltonian Path & Circuit

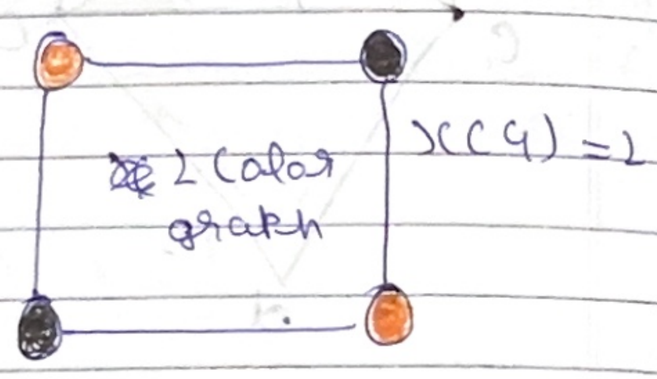
☆ Chromatic number of a graph

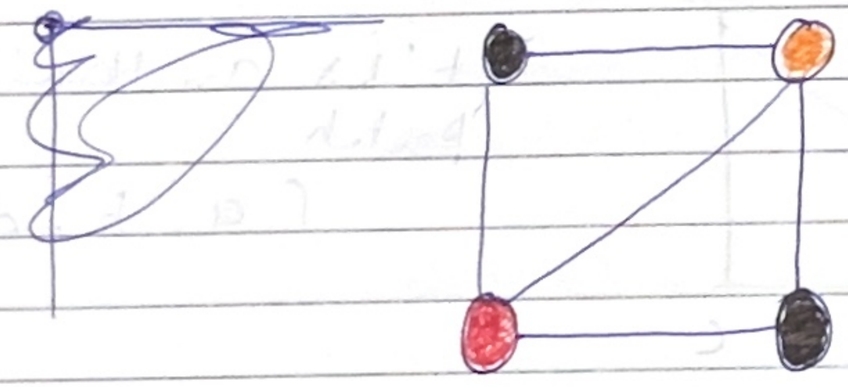
Graph Coloring
Coloring of Graph constitutes coloring Vertices / Edges of the graph.
Coloring all the vertices of a graph is the property that no two adjacent vertices have same colour.

Date : / /
Page No.

Chromatic Number: It is defined as the least no of Colors needed for Coloring the Graph.
denoted by $\chi(G)$ or K- Chromatic Graph.





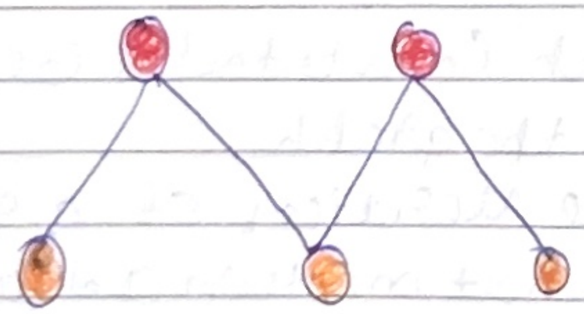


$\chi(G) = 3$

⇒ Every Bipartite Graph is 2 Colourable
$\chi(G) = 2$

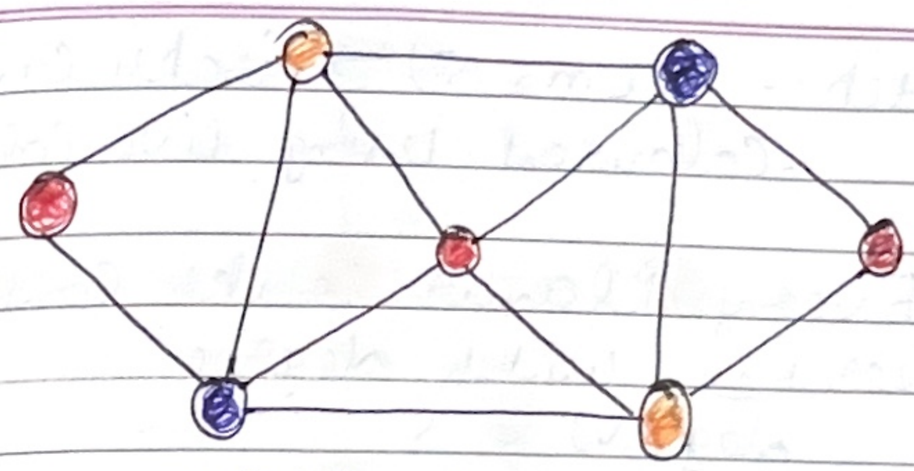
$V = \{v_1, v_2, v_3, v_4, v_5\}$

To vertices same set ko belong Karti ha Unka beach ma Edge nhi hogi



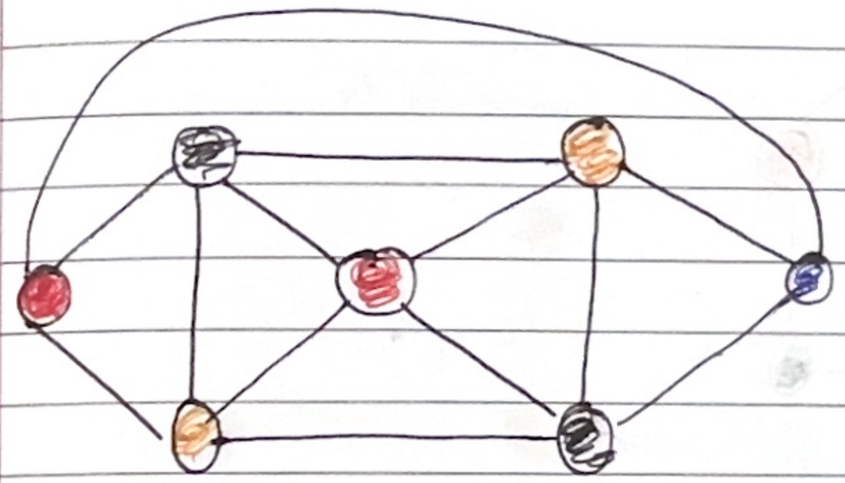
We can Color the Vertices of same set with one color.

Q



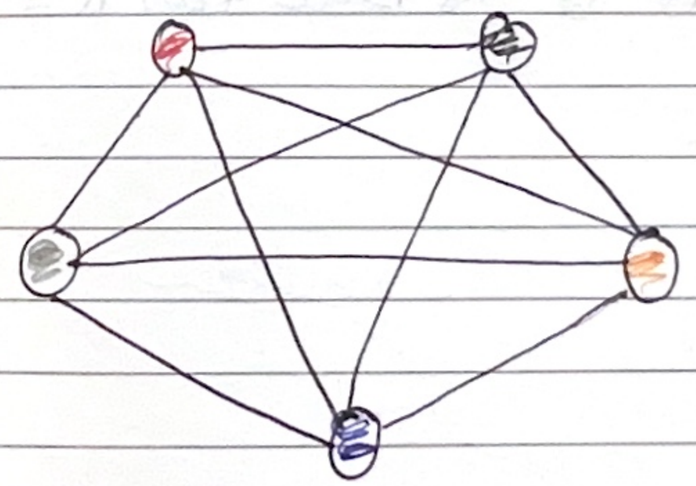
$\chi(G) = 3$

Q



$\chi(G) = 4$

Q



$\chi(G) = 5$

☆ 5 Color Theorem Proof

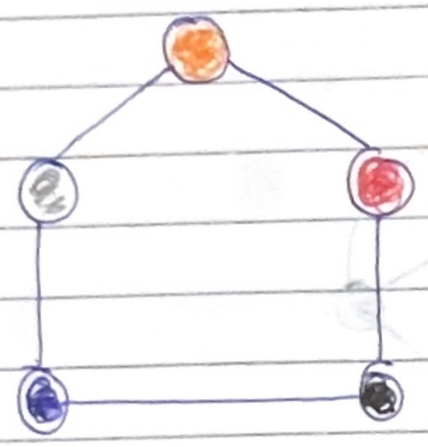
Theorem: Every Planar Graph with 'n' Vertices can be Colored Using atmost 5 Colors.

Proof:
Theorem by Mathematical Induction Method

Date : / /
Page No.

i) Basic Step :- P(n ≤ 5) → Graph can be Coloured using five colours.

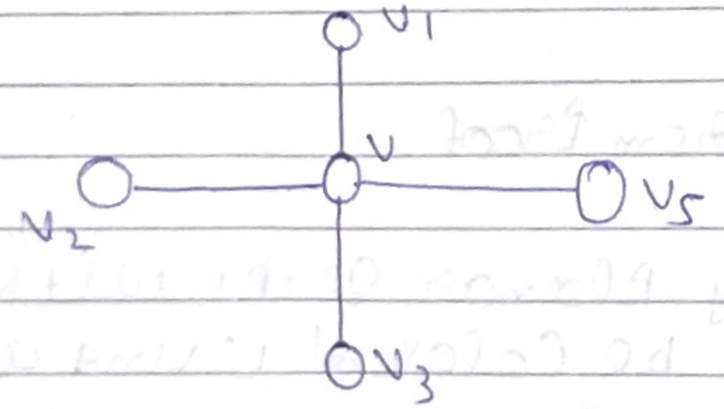
lemma : Every Planar Graph Contain a vertex with degree deg(v) ≤ 5



ii) Induction Step:
Let assume that it it true for k = 4

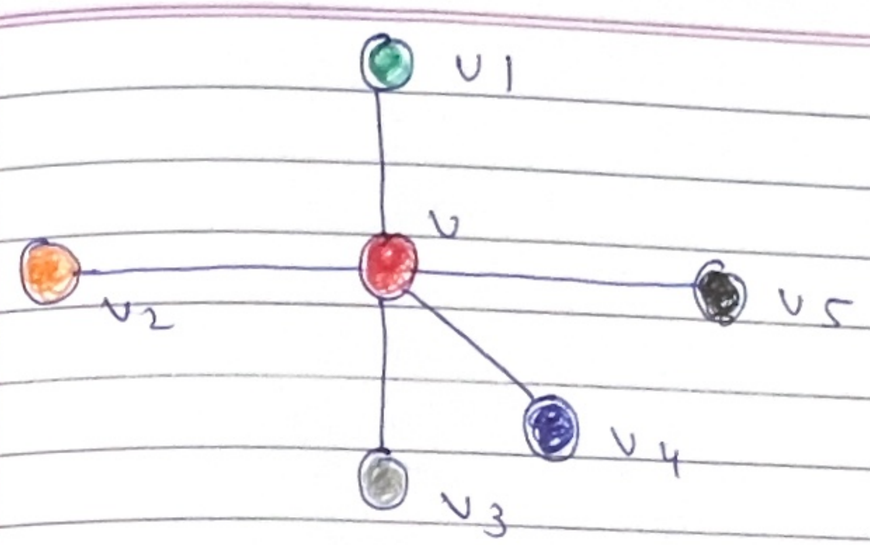
deg(v) ≤ 4

iii) Verify for k+1
deg(v) ≤ 5

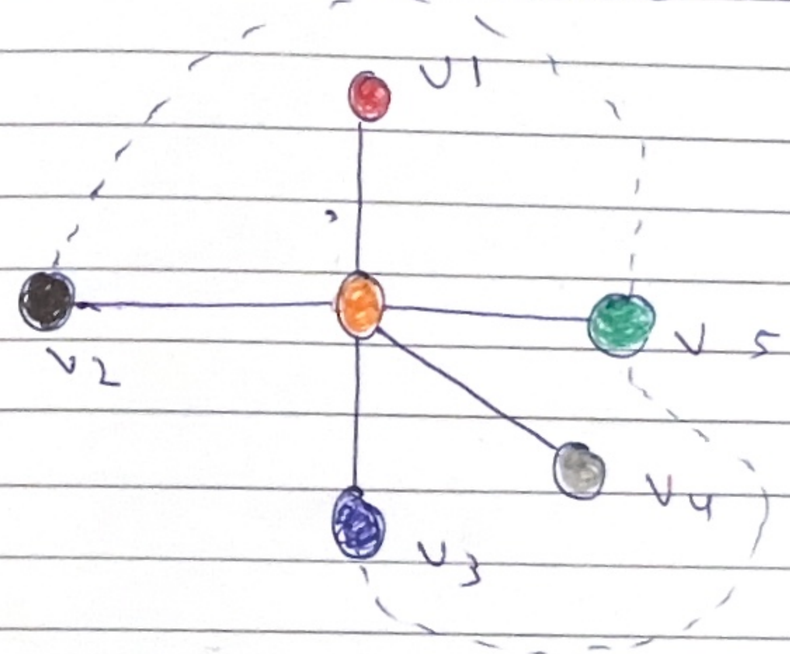


Case 1 : No direct edge

Date : / /
Page No.



Case 2 : Assume direct Edge



Hence Prove.