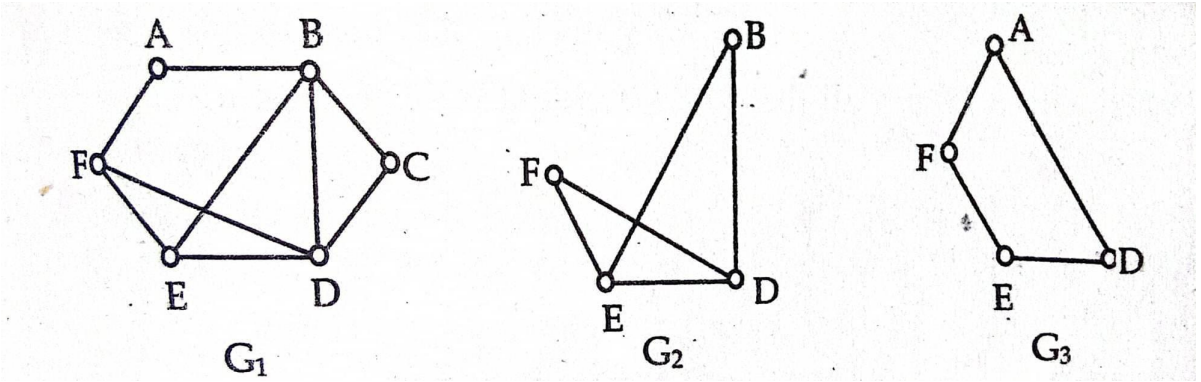
Graph Theory

1. **Sub Graph:** Let G=(V,E) be a graph with vertex set V(G) and edge set E(G) and H=(V,E) be a graph with vertex set V(H) and edge set E(H), then H is said to be subgraph of G If,

(i). All the vertex of H are in G.

(ii). All edges of H are in G.

(iii). Each edge of H has the same end points in H as in G.



Note: In above figures , G2 is a sub-graph of G1 but G3 is not a subgraph of G1.

2. Union and Intersection of a graph.

Union and Intersection of Graphs

Union

Given two graphs G_1 and G_2, their union will be a graph such that

$$V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$$

and $$E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$$

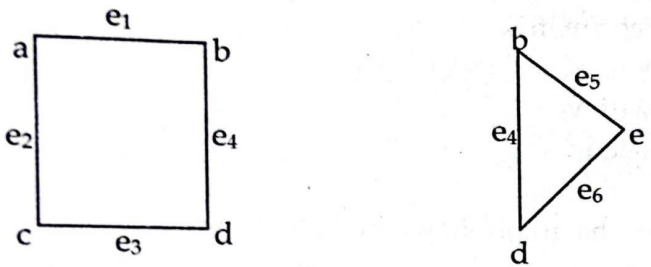
Intersections

Given two graphs G_1 and G_2, their intersection will be a graph such that

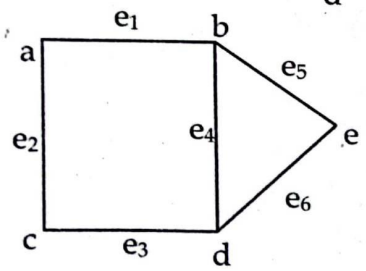
$$V(G_1 \cap G_2) = V(G_1) \cap V(G_2)$$

$$E(G_1 \cap G_2) = E(G_1) \cap E(G_2)$$

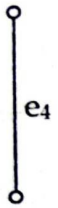
Consider two graphs G_1 and G_2 as



Then $G_1 \cup G_2$



and $G_1 \cap G_2$ is



3. Representation of a Graph.

- One of the representation method of Graph is Matrix representation, based on adjacency of vertices ,called ***adjacency matrix.***
- Another representation is based on incidence of vertices and edges called ***incidence matrix.***

1. Adjacency Matrix:

- Let G=(V,E) be a graph with ‘n’ vertices V1,V2,V3,…….Vn. The adjacency matrix of G with respect to given ordered list of vertices is a n*n matrix denoted by A(G), such that

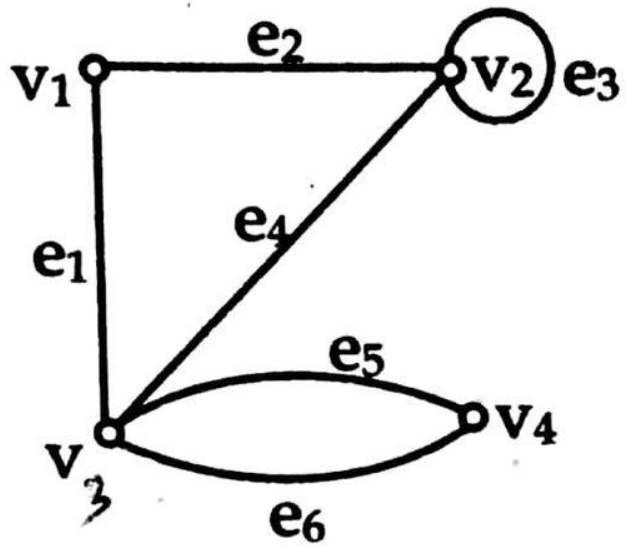
a_{ij} ={ 0 if there is no edge between the vertices.

{1 if there is an edge between the vertices.

{ k if there are k>=2 edges between the vertices.

Example

Find the adjacency matrix to represent the graph shown in figure given below.



Solution

Since G has four vertices, adjacence matrix A(G) will be a 4 × 4 matrix.

$$A(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix} \end{array}$$

2. Incidence Matrix

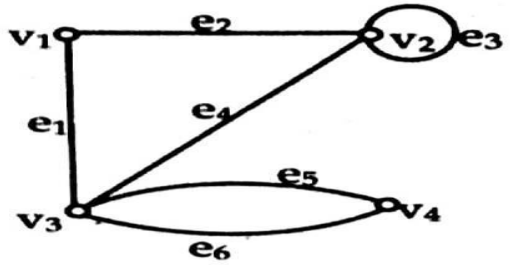
Incidence Matrix

Let G be a graph with vertices $v_1, v_2, \ldots, v_m$ and edges $e_1, e_2, \ldots e_n$. The incidence matrix I(G) of graph G is a $m \times n$ matrix with $I(G) = (m_{ij})_{m \times n}$, where

$$m_{ij} = \begin{cases} 1 \text{ if } e_j \text{ is incident with } v_i \\ 0 \text{ if } e_j \text{ is not incident with } v_i \\ 2 \text{ if } v_i \text{ is the end of the loop} \end{cases}$$

Example

Find the incidence matrix to represent the graph shown in figure below



Solution

Since G ahs four vertices and six edges, incidence matrix I(G) will be a 4 × 6 matrix.

$$I(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

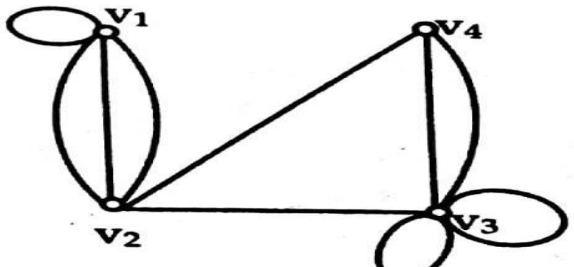
Example

Draw the graph G corresponding to the following adjacency matrix.

$$A = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

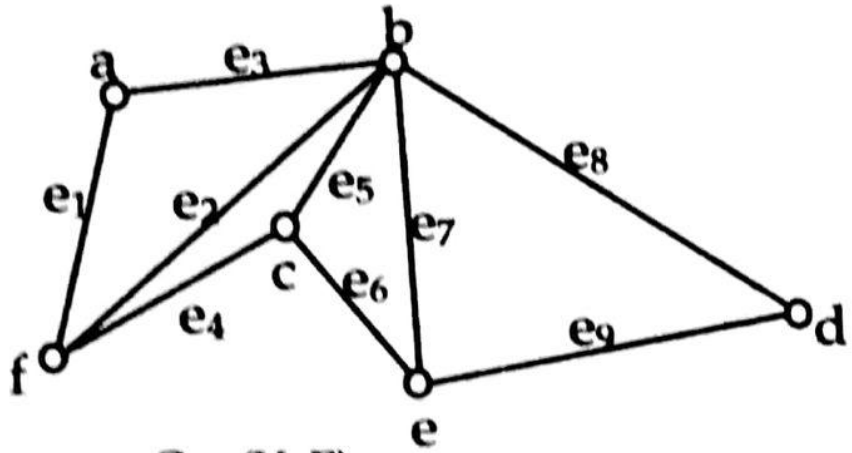
Solution

Since A is a 4 – square matrix, G has four vertices say v_1, v_2, v_3 and v_4. Then G is



Example

Find the adjacency matrix and incidence matrix of the graph given below.



G = (V, E)

Solution

Let the order of the vertices be a, b, c, d, e, f and edges order be $e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9$

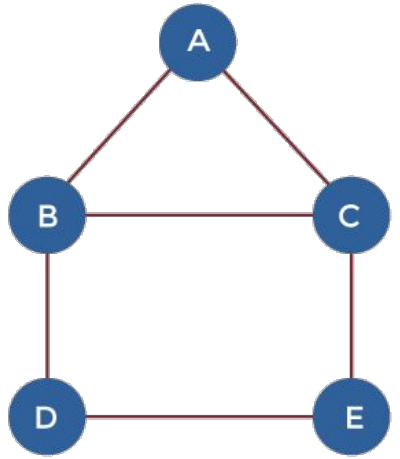
$$A(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad I(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Graph Connectivity

1. **Walk :** A walk can be defined as a sequence of edges and vertices of a graph. When we have a graph and traverse it, then that traverse will be known as a walk. In a walk, there can be repeated edges and vertices. The number of edges which is covered in a walk will be known as the Length of the walk.

So for a walk, the following two points are important, which are described as follows:

- ***Edges can be repeated***
- ***Vertex can be repeated***



In the above graph, there can be many walks, but some of them are described as follows:

1. A, B, C, E, D (Number of length = 4)
2. D, B, A, C, E, D, B, C (Number of length = 7)
3. E, C, B, A, C, E, D (Number of length = 6)

❑ Types of Walks

- There are two types of the walk, which are described as follows:

1. Open walk
2. Closed walk

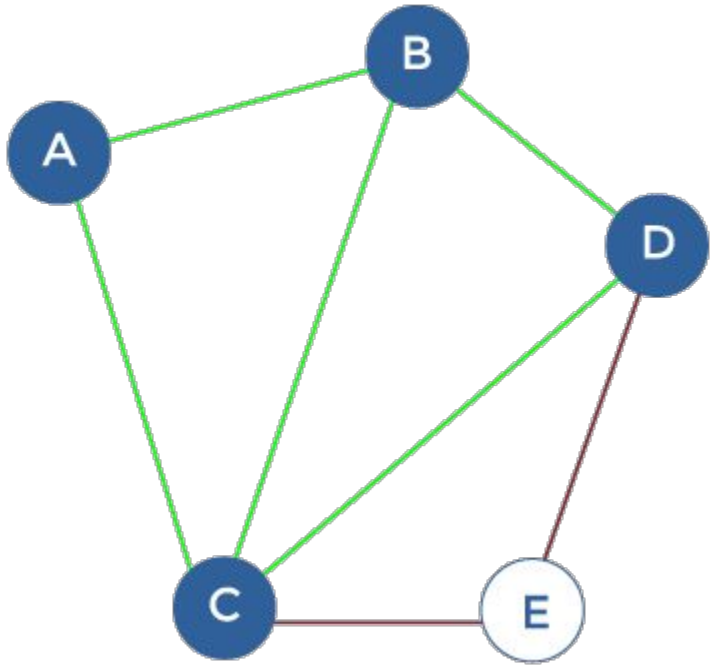
- **Open Walk:**

A walk will be known as an open walk in the graph theory if the vertices at which the walk starts and ends are different. That means for an open walk, the starting vertex and ending vertex must be different.

- **Closed Walk:**

A walk will be known as a closed walk in the graph theory if the vertices at which the walk starts and ends are identical. That means for a closed walk, the starting vertex and ending vertex must be the same.

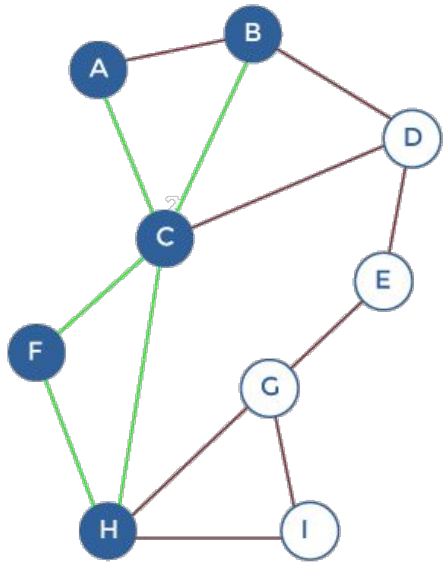
- In this graph, there is also a closed walk and an open walk, which are described as follows:



- Closed walk = A, B, C, D, E, C, A
- Open walk = A, B, C, D, E, C

2. Trails

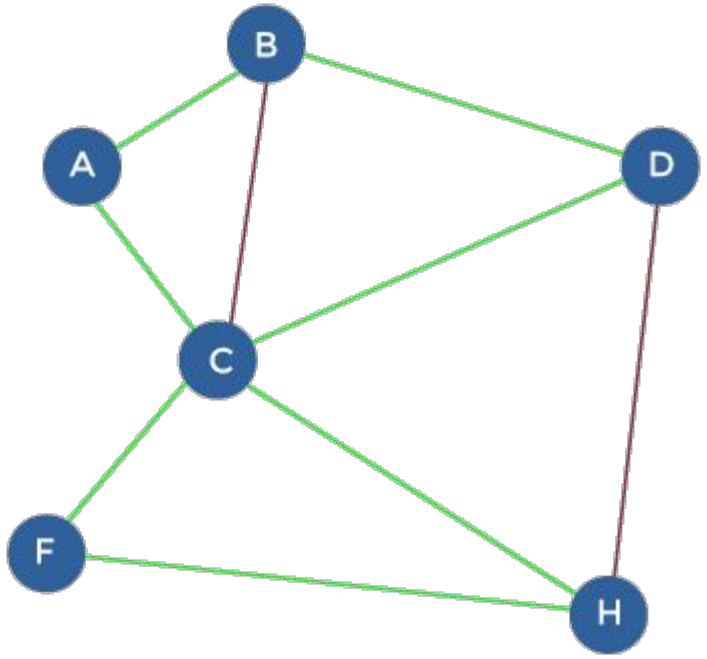
- A trail can be described as a walk where ***no edge is allowed to repeat***. In the trails, the ***vertex can be repeated.***



- In the above graph, there is a tail and closed tail, which is described as follows:
- Open Tail = A, C, H, F, C, B
- Closed tail = A, C, H, F, C, B, A

3. Circuit

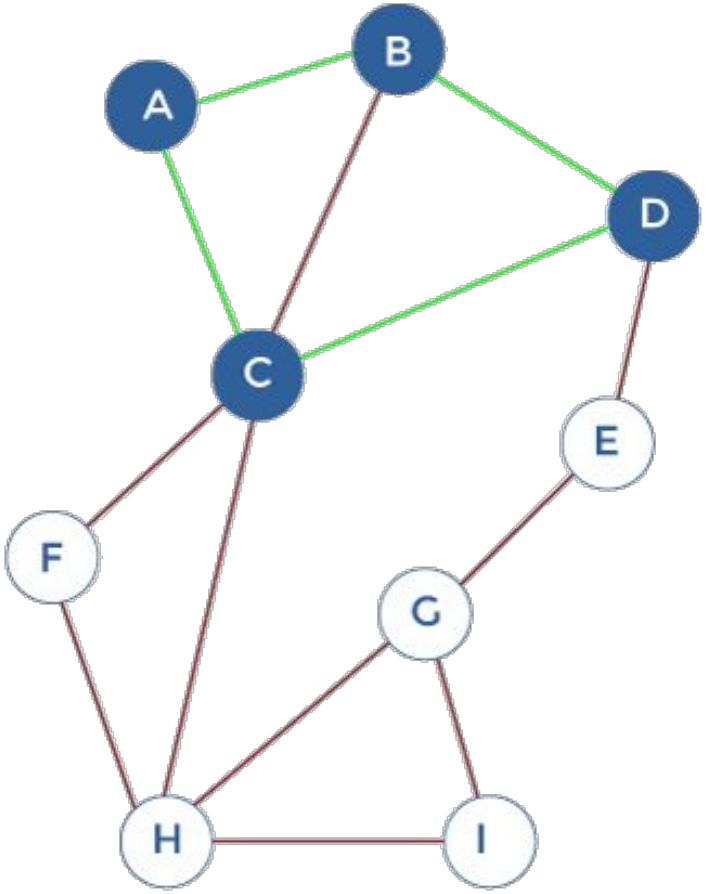
- A closed trail in the graph theory is also known as a circuit. So for a circuit, the following two points are important, which are described as follows:
- Edges cannot be repeated
- Vertex can be repeated



Circuit: A, B, D, C, F, H, C, A

4. Cycle:

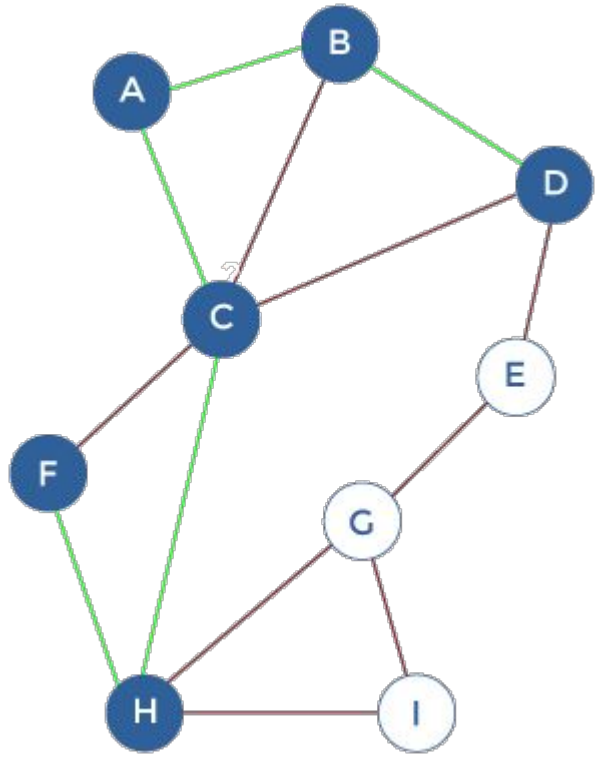
- In cycle, neither edges nor vertices are allowed to repeat. There is a possibility that only the starting vertex and ending vertex are the same in a cycle. So for a cycle, the following two points are important, which are described as follows:
- Edges cannot be repeated
- Vertex cannot be repeated



Cycle: A, B, D, C, A

4. Path:

- A path is a type of open walk where neither edges nor vertices are allowed to repeat. There is a possibility that only the starting vertex and ending vertex are the same in a path. So for a path, the following two points are important, which are described as follows:
- Edges cannot be repeated
- Vertex cannot be repeated

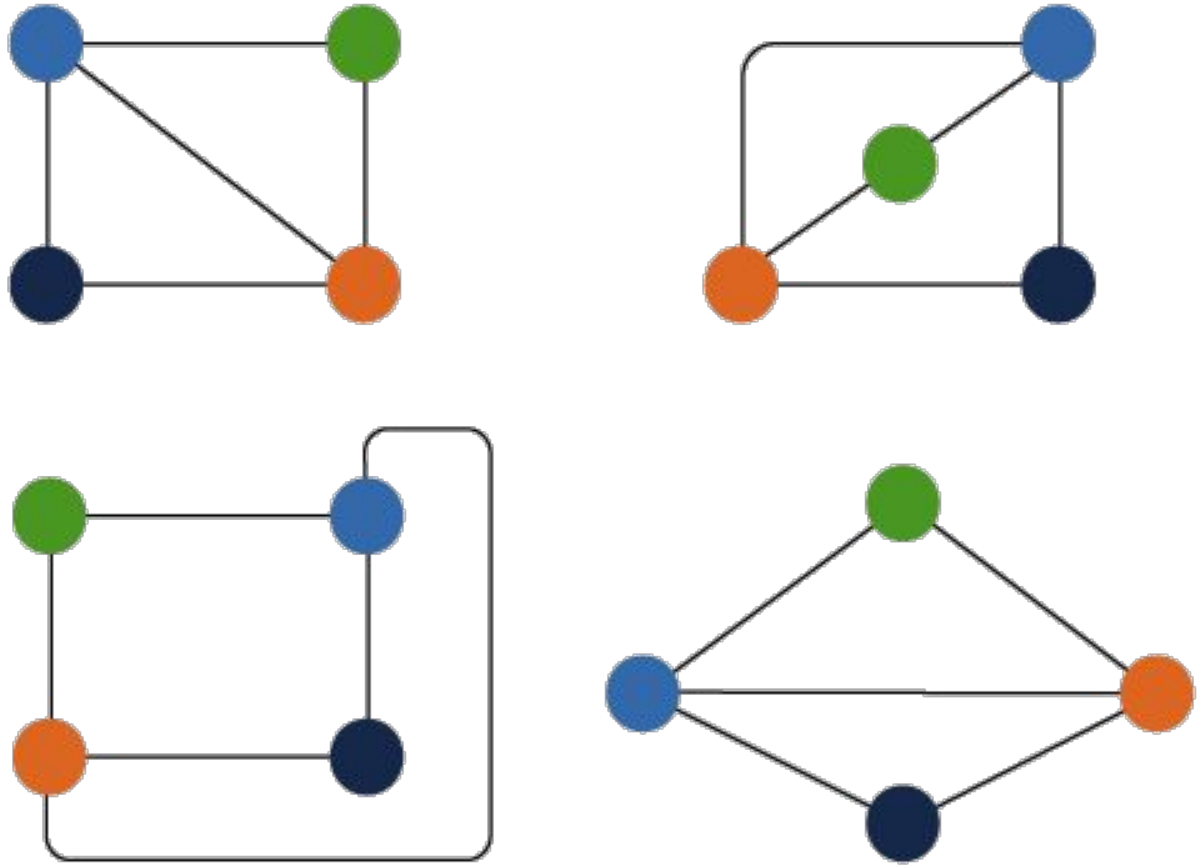


- Path: F, H, C, A, B, D (Open Path)

  : A, C, D, B, A (Closed Path □ Cycle)

❑ Isomorphism of a Graph

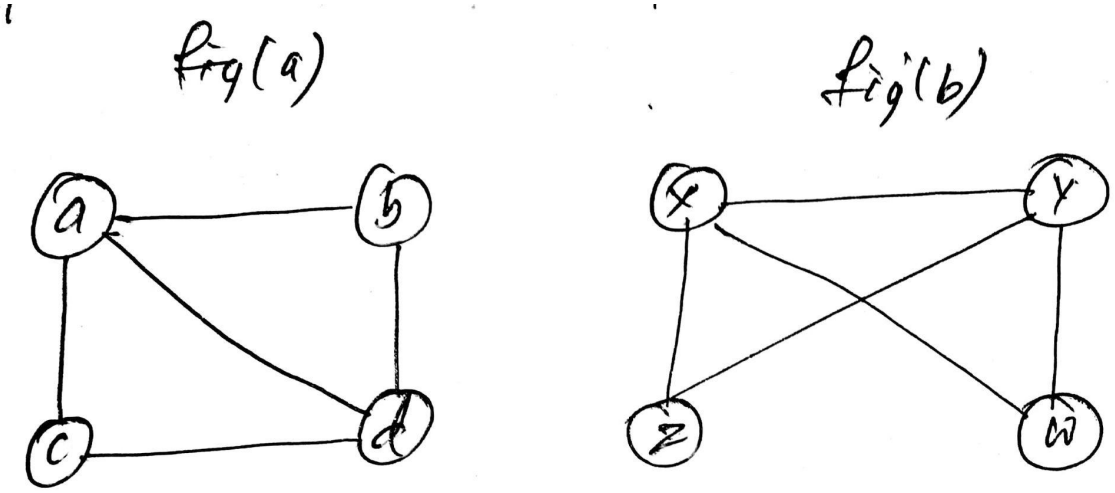
The isomorphism graph can be described as a graph in which a single graph can have more than one form. That means two different graphs can have the same number of edges, vertices, and same edges connectivity. These types of graphs are known as ***isomorphism graphs.*** The example of an isomorphism graph is described as follows:



❑ Conditions for Graph isomorphism

- Any two graphs will be known as isomorphism if they satisfy the following four conditions:

1. There will be an equal number of vertices in the given graphs.
2. There will be an equal number of edges in the given graphs.
3. There will be an equal amount of degree sequence in the given graphs.
4. Identical Mapping.



- **Condition1**: No.of vertices: fig(a)= 4= fig(b).
- **Condition 2**: No. of edges: fig(a)=5=fig(b).
- **Condition 3**: Degree sequence: In fig(a): a=3, b=2, c=2, d=3.(Dec:3,3,2,2)

  : In fig(b): x=3,z=2, w=2, y=3.(Dec: 3,3,2,2)
- **Condition 4:** Mapping : a(3)□x(3)

  :b(2)□w(2)

  :c(2)□z(2)

  :d(3)□y(3)

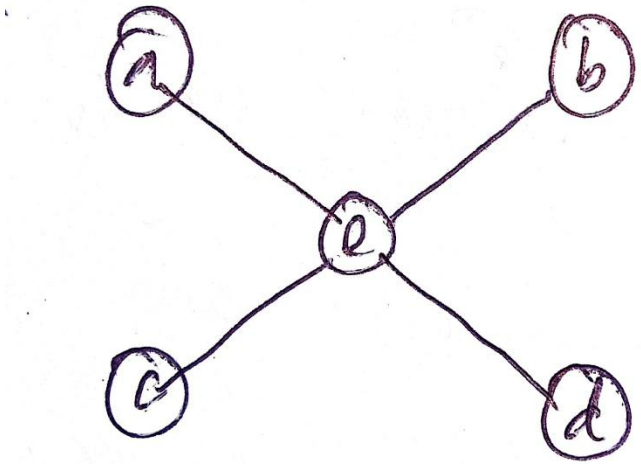
Hence, the above graphs are isomorphic graphs.

❑ Euler and Hamiltonian Graphs

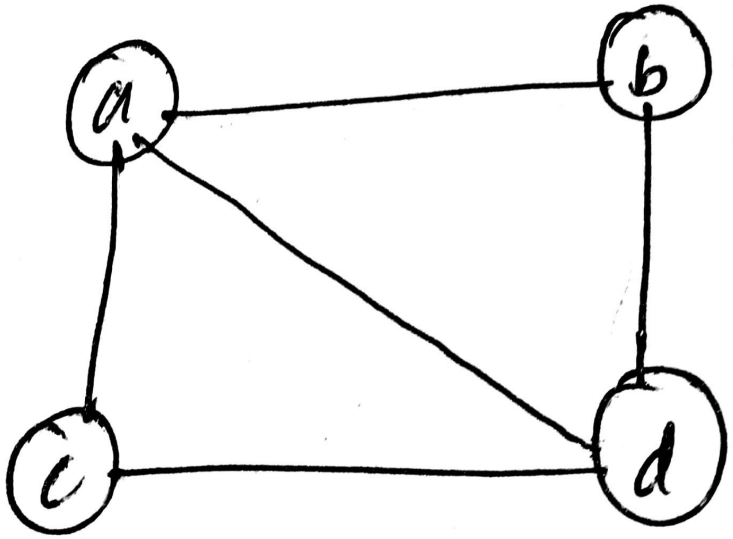
1. **Euler Graph (Euler path+ Euler circuit)**

a. **Euler Path: We cover every edges without any repetition of edges.**

b. **Euler circuit: We start with any vertex, cover every edges and reach to the starting point without repeating edges.**

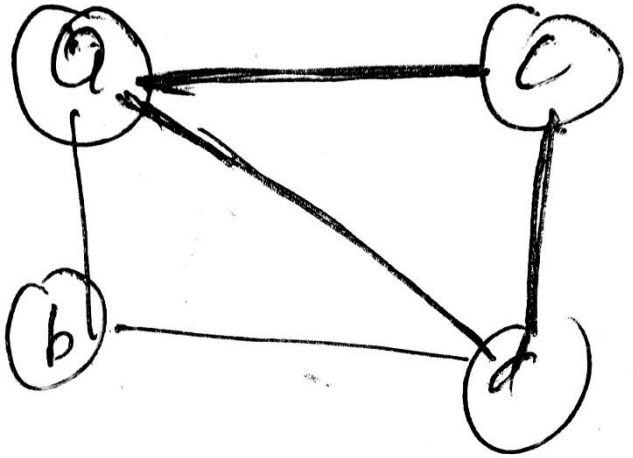


Lets travel: **In this graph it is not possible to cover every edges without repeating edges so, it is not a Euler path also means that it is not a Euler circuit . Hence it is not a Euler Graph.**

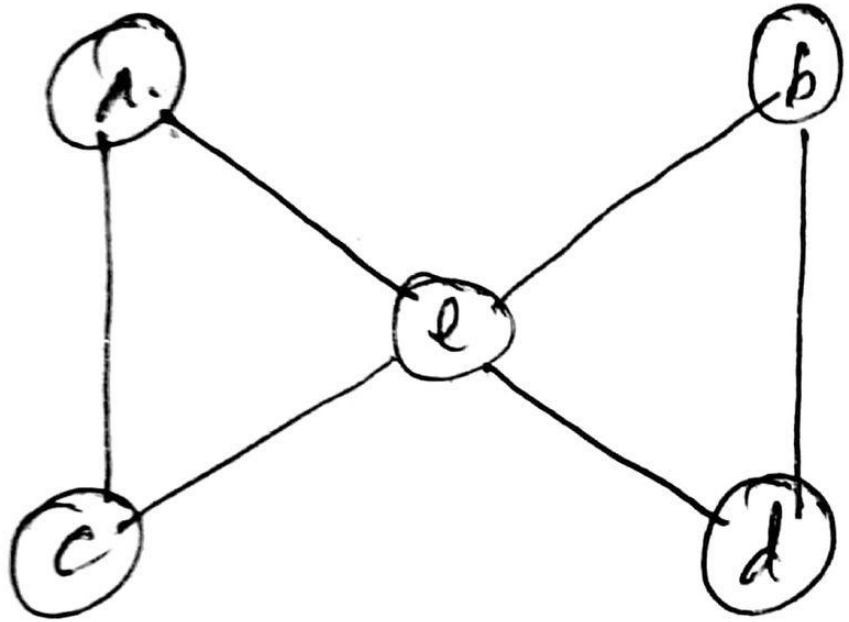


- Lets travel: a□b□d□a□c□d
- Here, we cover every edges, hence Euler path.
- But starting and end vertex not same(not possible), hence it is not Euler circuit.

❑ Trap case of Euler path: c□d□a□c□Trap



q3



let's Travel :-

$a \rightarrow e \rightarrow d \rightarrow b \rightarrow e \rightarrow c \rightarrow a$

Eular circuit ✓

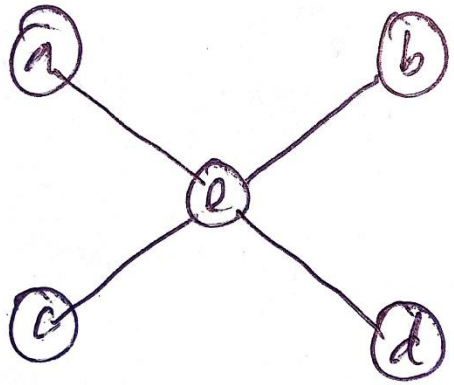
Eular path ✓

Eular graph ✓

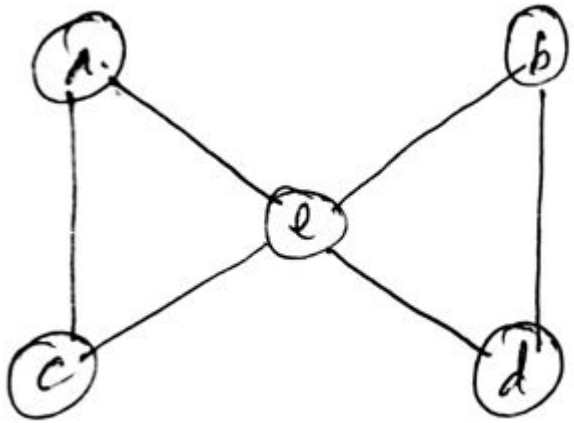
2. Hamiltonian Graph (Ham.path+Ham.circuit)

a. **Hamiltonian Path:** We cover every vertices without repetition of vertex.

b. **Hamiltonian Circuit:** We cover every vertex without repetition but start and end vertex should be same(repeated).

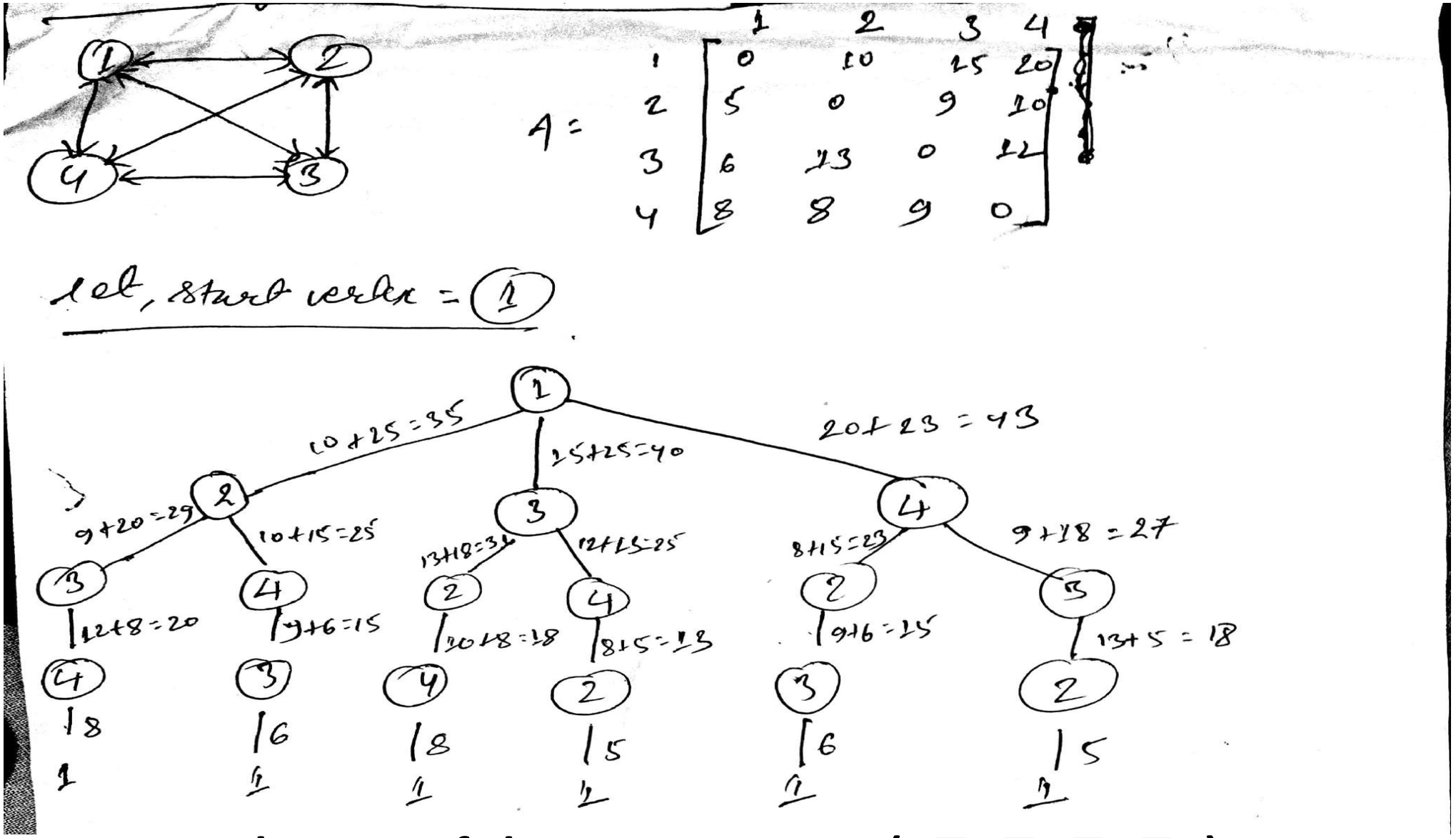


In this graph it is not possible to cover every vertex without repeating so, it is not a Ham. path also means that it is not a Ham circuit . Hence it is not a Ham. Graph.



- Lets travel: a□c□e□d□b (every vertex covered so Ham.Path)
- But not Ham.circuit because start and end vertex are not identical(not possible).
- Hence ,the above graph is not Ham.Graph.

❑ Travelling Salesman Problem(TSP):

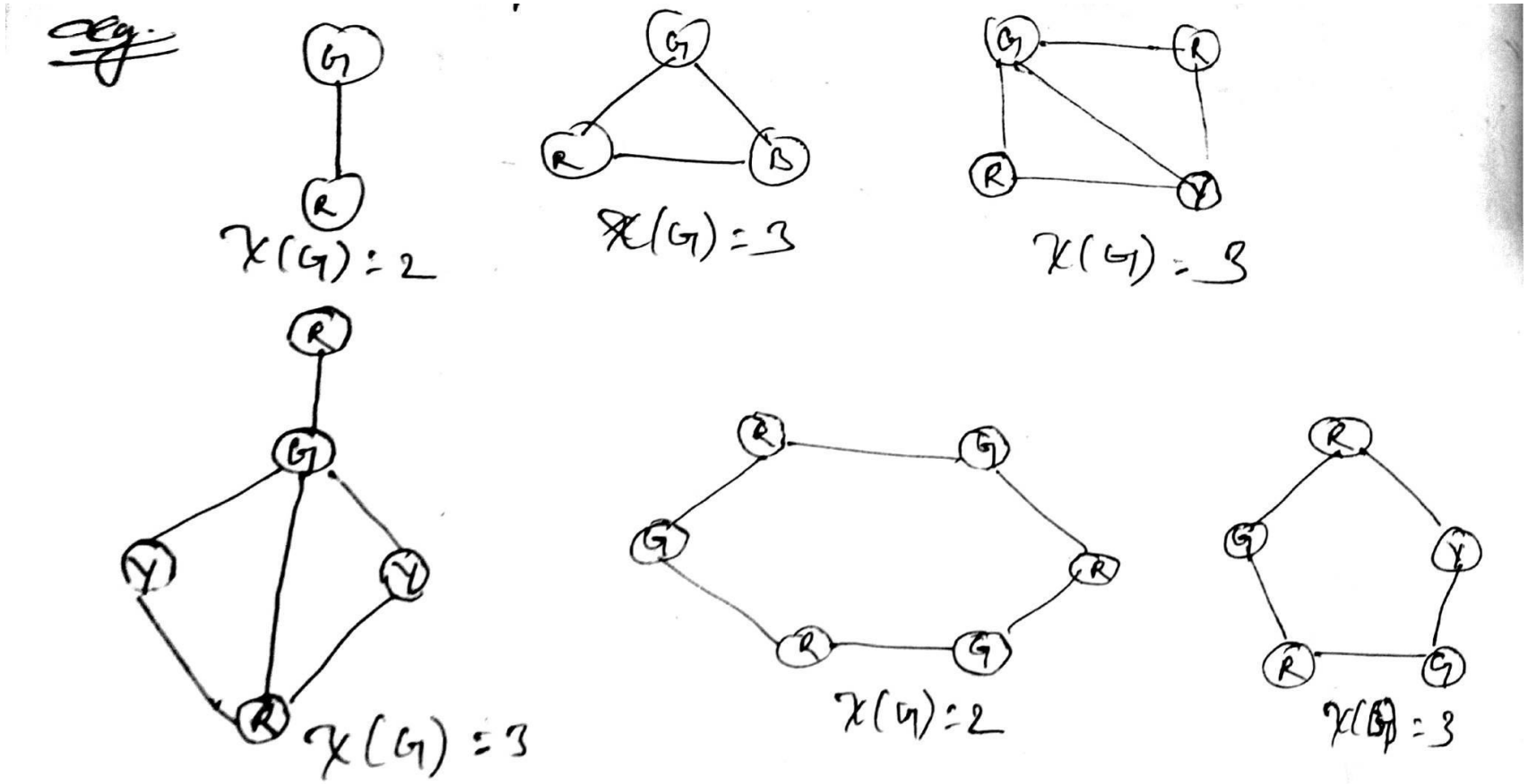


Hence, The cost of shortest Route: 35(1□2□4□3□1)

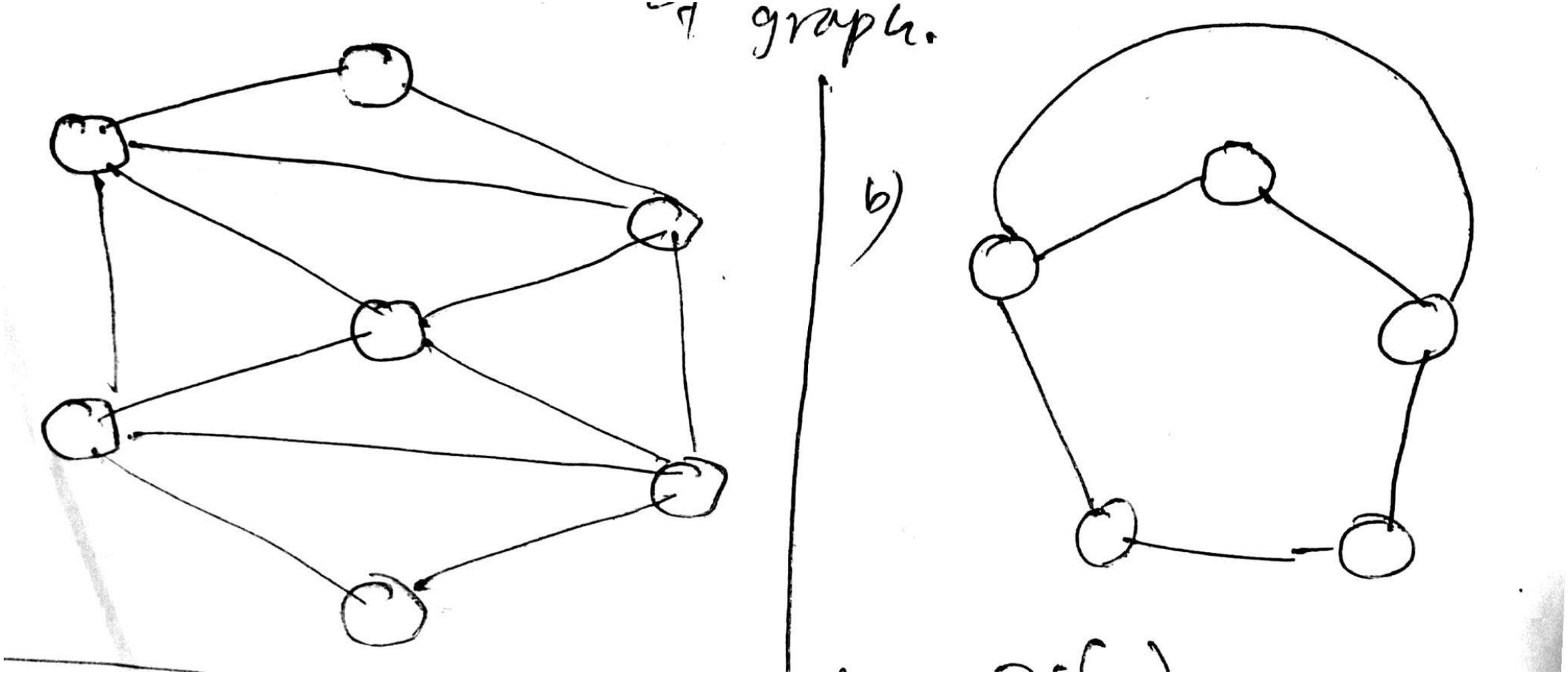
❑ Graph coloring

Painting all vertices of a graph with colors such that no two adjacent vertices have the same color is called coloring of graph.

Chromatic Number: It is the least no.of colors required for coloring of a graph ‘G’ is called it’s chromatic number. It is denoted by Xi(G).

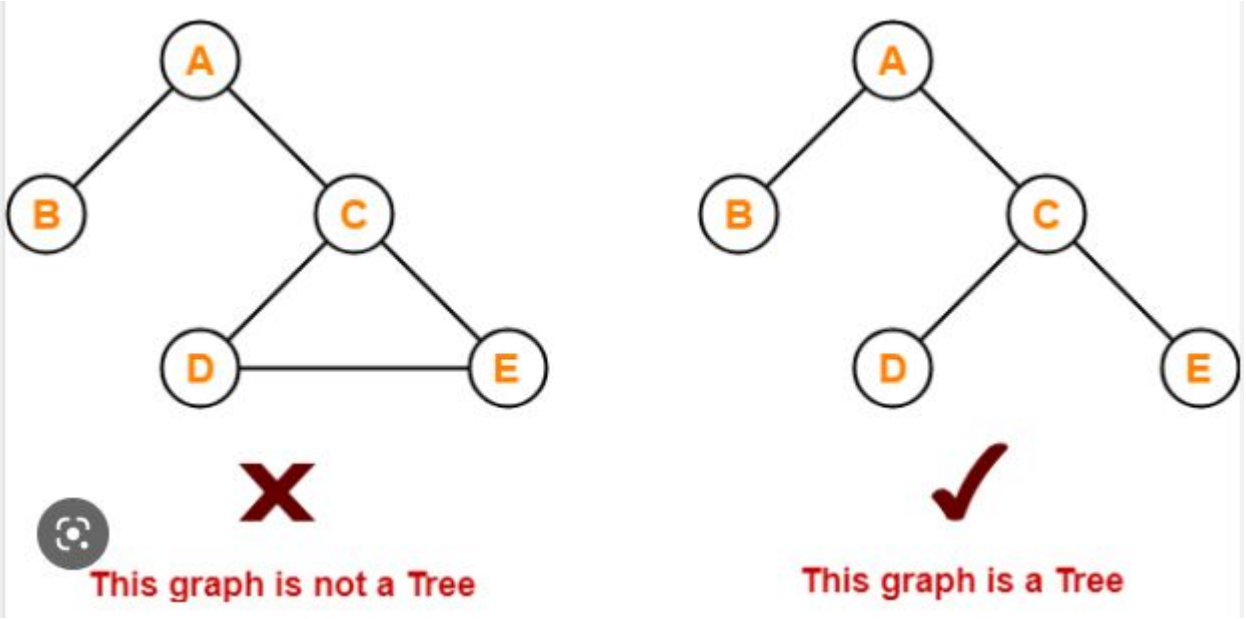


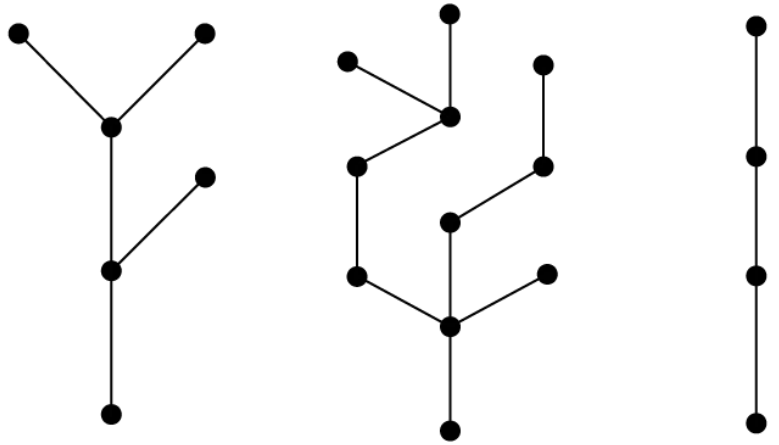
❑ *Find the chromatic number of a graph.*



❑ Tree

- A tree is a connected graph without any cycle.

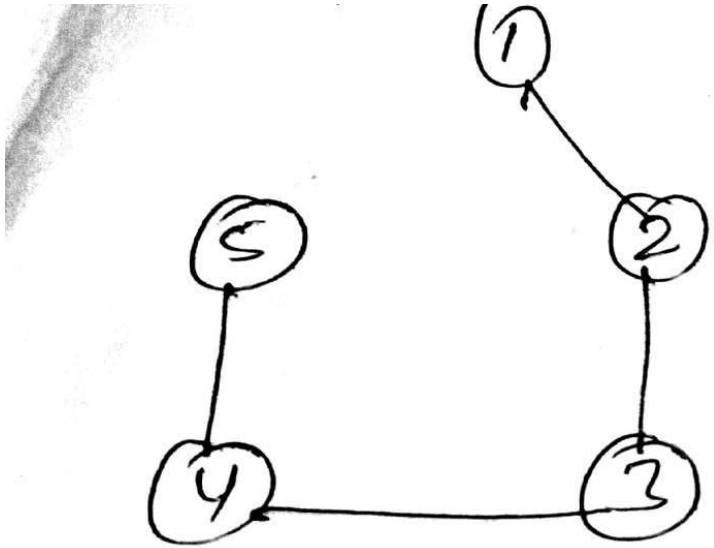




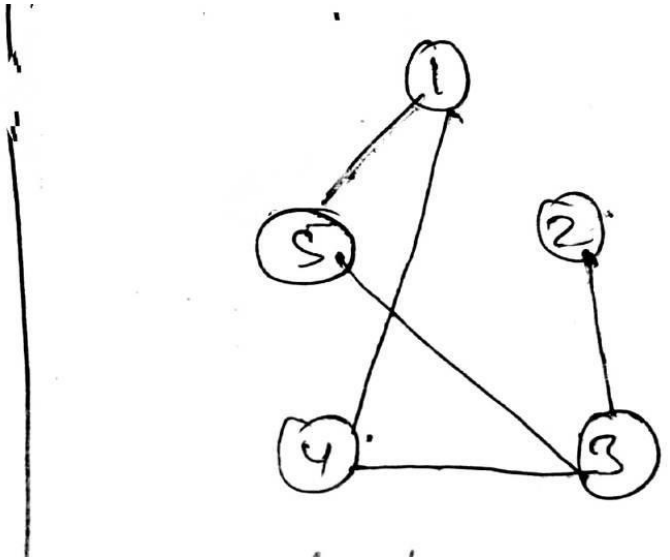
Note: A tree with ‘n’ vertices has (n-1) edges(No cycle allowed).

PROOF:

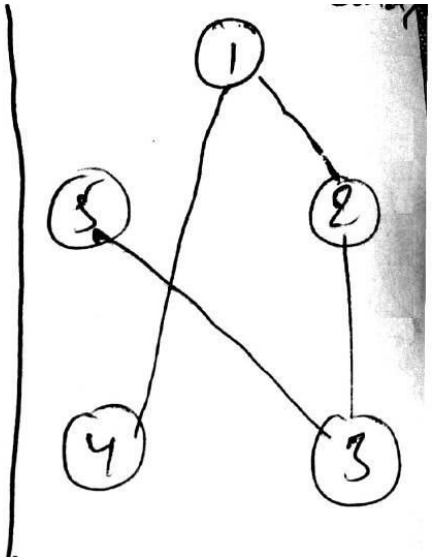
Case :min(no.of edges)=4



Case : cycle

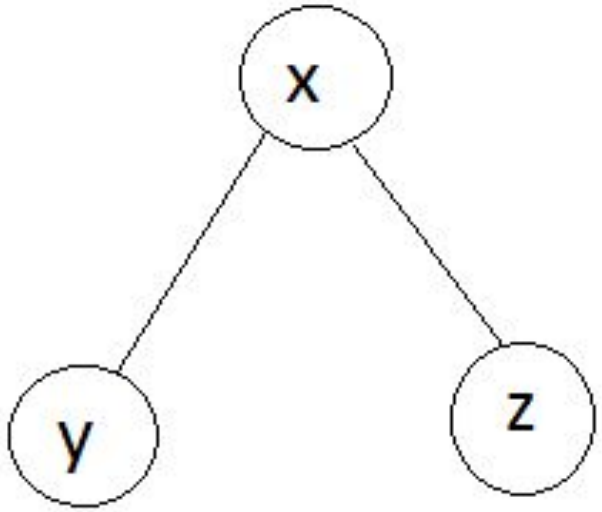


Case :max(no.of edges)=4



❑ Tree Traversal

1. Pre-order- Root Left Right
2. In-order- Left Root Right
3. Post-order- Left Right Root



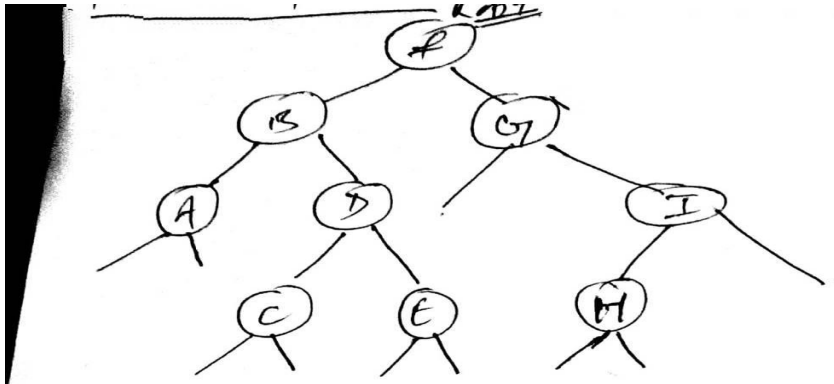
1.Pre-order□ xyz

2. In-order□ yxz

3. Post-order□yzx

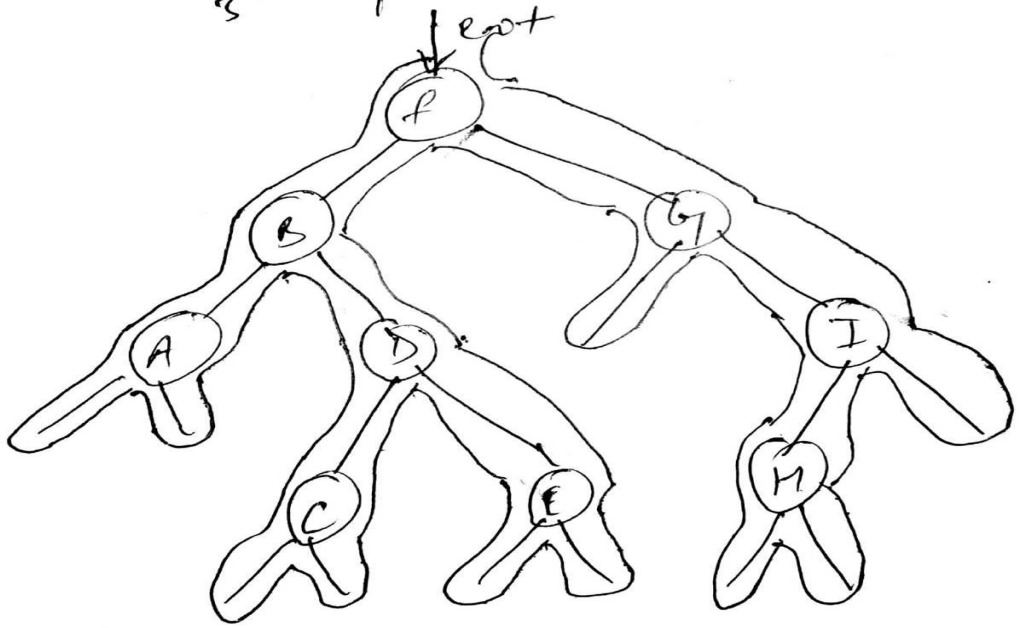
Example2:

First ensures the two children's of each node to make BST.(i.e. make dummy children)



Solution,

1st preorder : F B A D C E G I H
2nd Inorder : A B C D E F G H I
3rd postorder : A C E D B H I G F



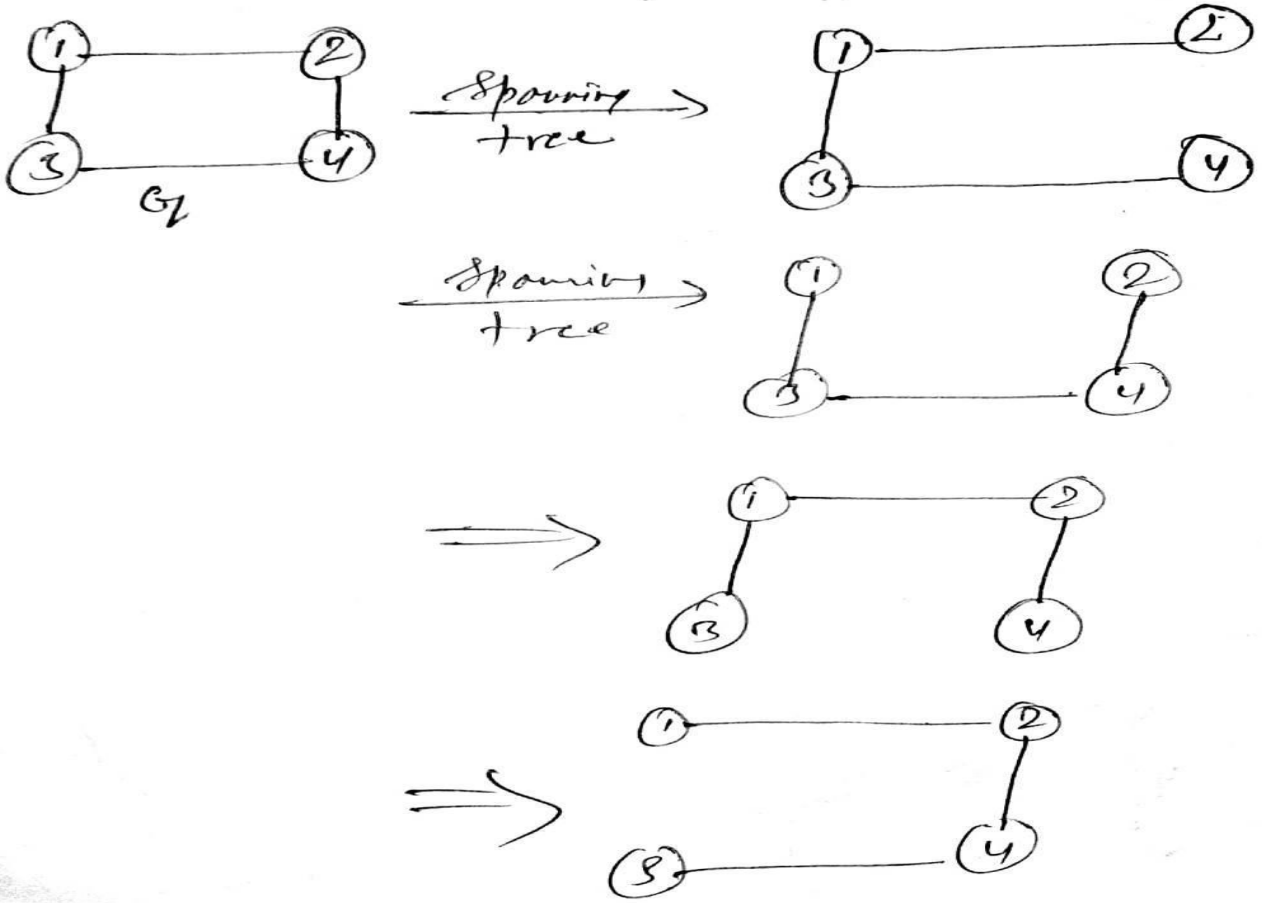
❑ Spanning Tree:

Spanning Tree

— A connected subgraph 'S' of Graph G(V, E) is said to be spanning iff

a) 'S' should contain all vertices of 'G'
b) 'S' should contain (|V| − 1) edges.
c) 'S' should not contain a cycle because it's a tree.

Example 1: let's assume Graph G (V, E)

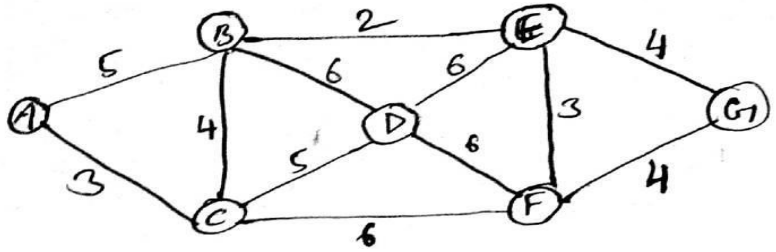


❑ *Minimum Spanning Tree(MST)*

Minimum Spanning tree - Minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of the spanning tree.

Minimum Spanning Tree (MST)

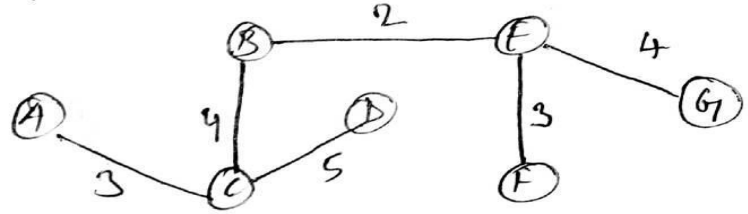
1) Kruskal Algorithm:



Solution

Edges	Weight (Increasing order)	Status
~~A-B~~ B-E	2	✓
A-C	3	✓
E-F	3	✓
B-C	4	✓
E-G1	4	✓
F-G1	4	✓
C-D	5	✗
A-B	5	✓
B-D	6	✗
D-E	6	✗
D-F	6	✗
C-F	6	✗
		✗

MST: property

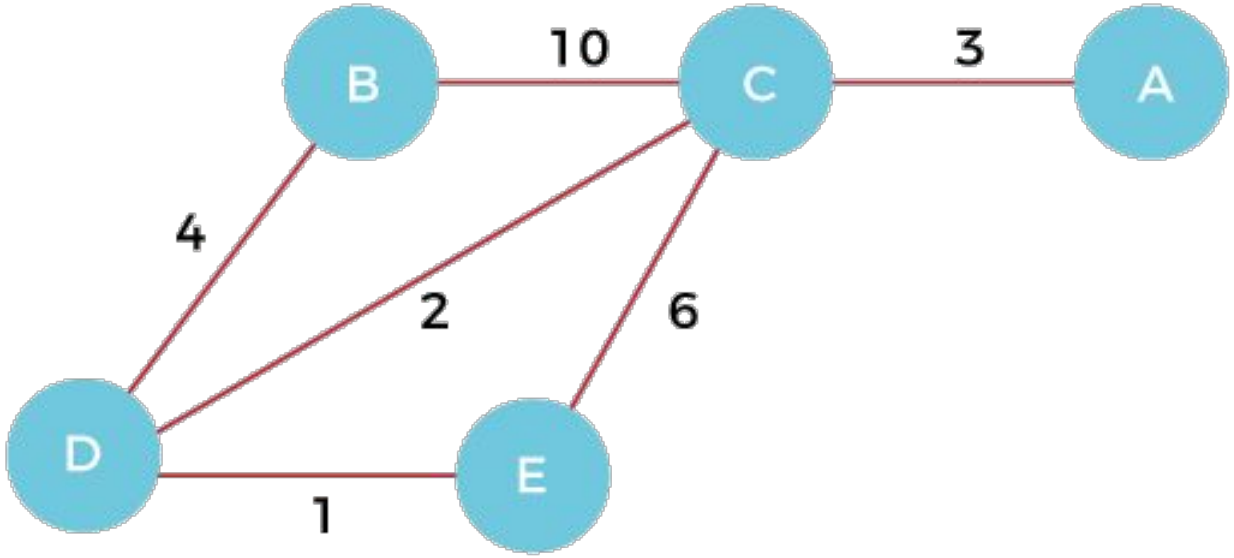


properties	
connected	✓
All vertex	✓
No cycle	✓
Min. cost	✓

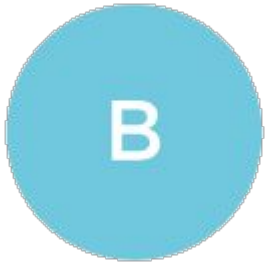
- **Prim's Algorithm** is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Example of prim's algorithm

- Now, let's see the working of prim's algorithm using an example. It will be easier to understand the prim's algorithm using an example.

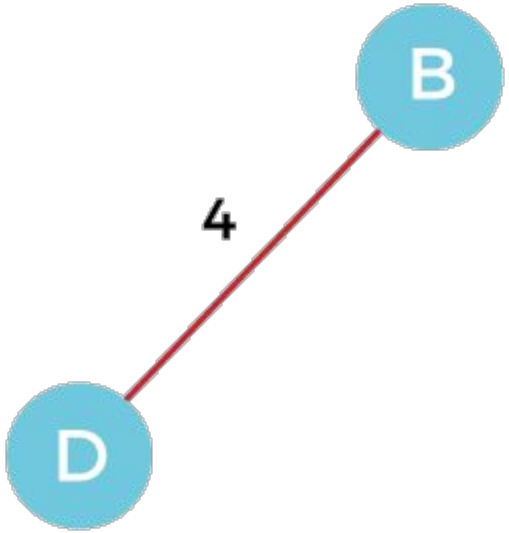


- ***Step 1*** **-** First, we have to choose a vertex from the above graph. Let's choose B.

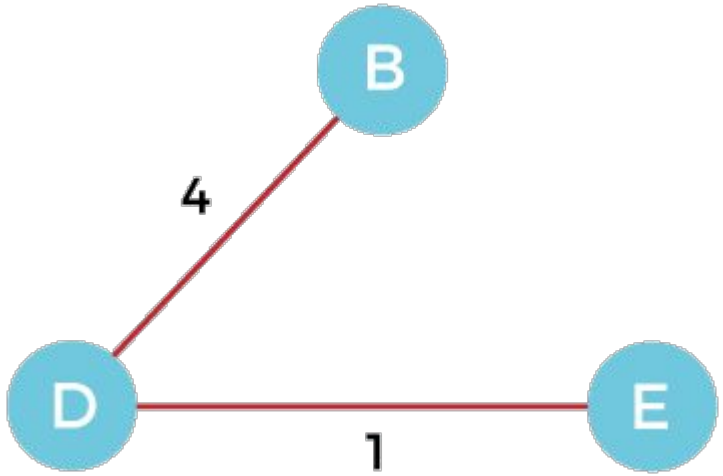


- ***Step 2*** **-** Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge *BD* has the minimum weight. So, add it to the MST.

Note: Remaining edge: *BC(10)*

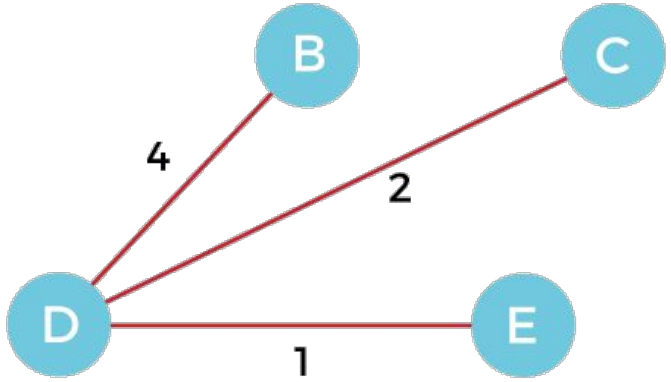


- ***Step 3*** **-** Now, current node is ‘D’ .In this case, the possible the edges *DE(1)* and *DC(2)* are such edges + old edge in account BC(10). Now select minimum weighted edge and add to MST So, selected edge is *DE(1)* and add it to the MST.
- Note: Remaining edge: *BC(10) , DC(2)*

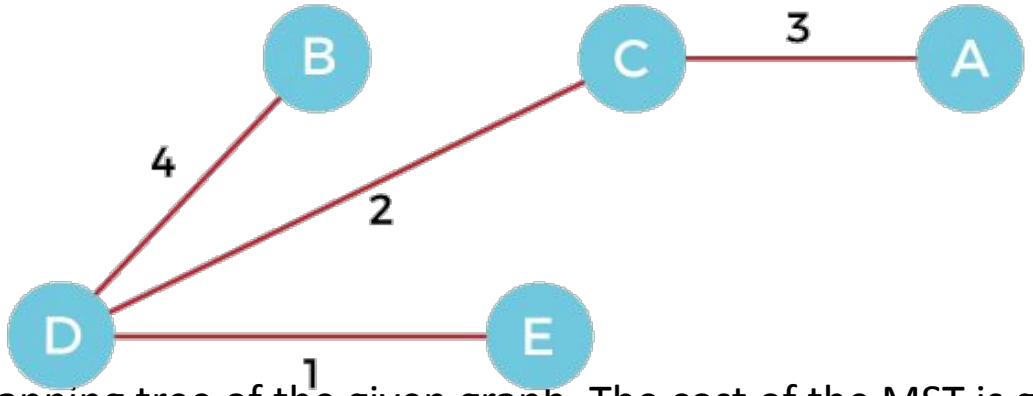


- ***Step 4*** **-** Now, current node ‘E’ ,possible travel is ***EC(6)***+ old edges *BC(10) , DC(2)* select the edge *DC(2)* which is minimum , and add it to the MST.

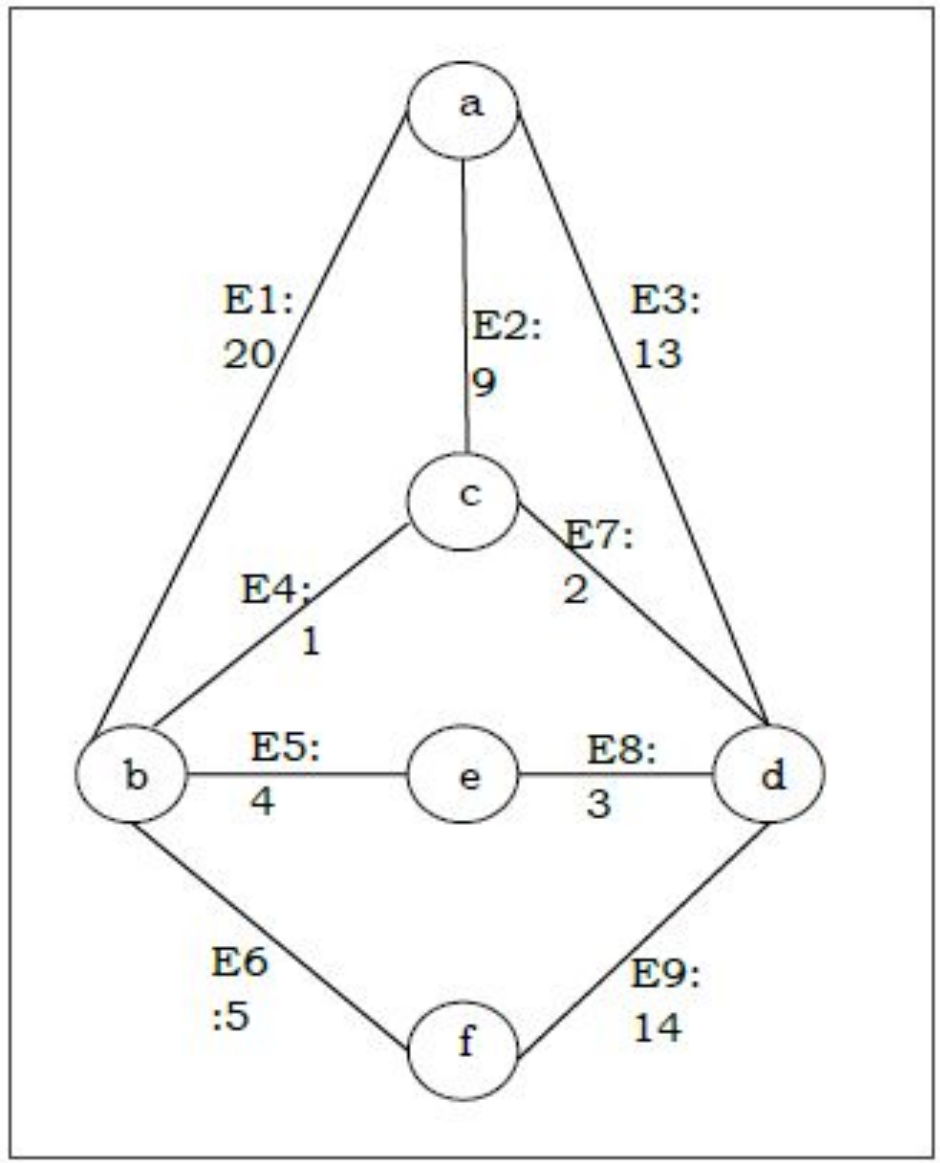
Note: Remaining edge: *EC(6), BC(10)*



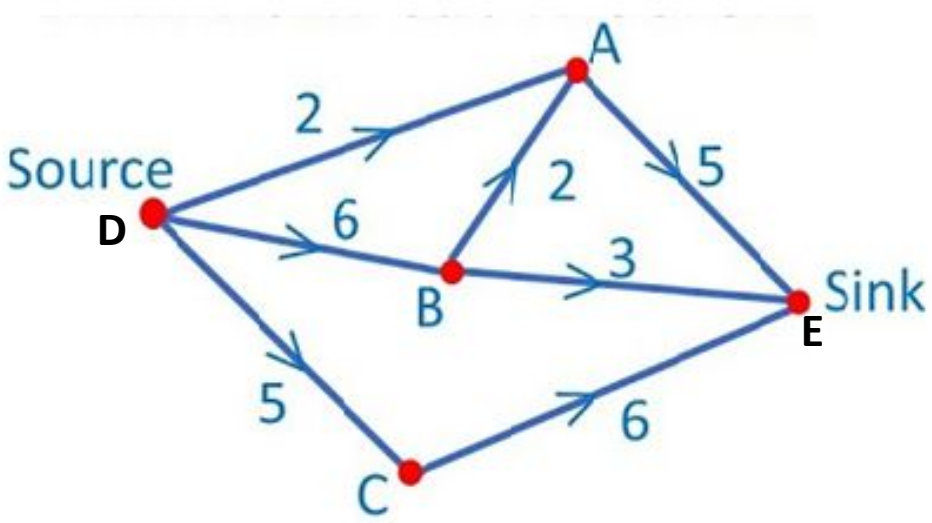
- **Step 5 -** Now, current node is ‘C’ , possible travel is CA(3)+Remaining edge: *EC(6), BC(10)* choose the edge CA(3). Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.



- So, the graph produced in step 5 is the minimum spanning tree of the given graph. The cost of the MST is given below -
- Cost of MST = 4 + 2 + 1 + 3 = 10 units.


a
E1:
20
E2:
9
E3:
13
c
E7:
2
E4:
1
b
E5:
4
e
E8:
3
d
E6
:5
E9:
14
f

❑ Maximum Flow Minimum cut Theorem [Ford Fulkerson Algorithm]



Terms used:

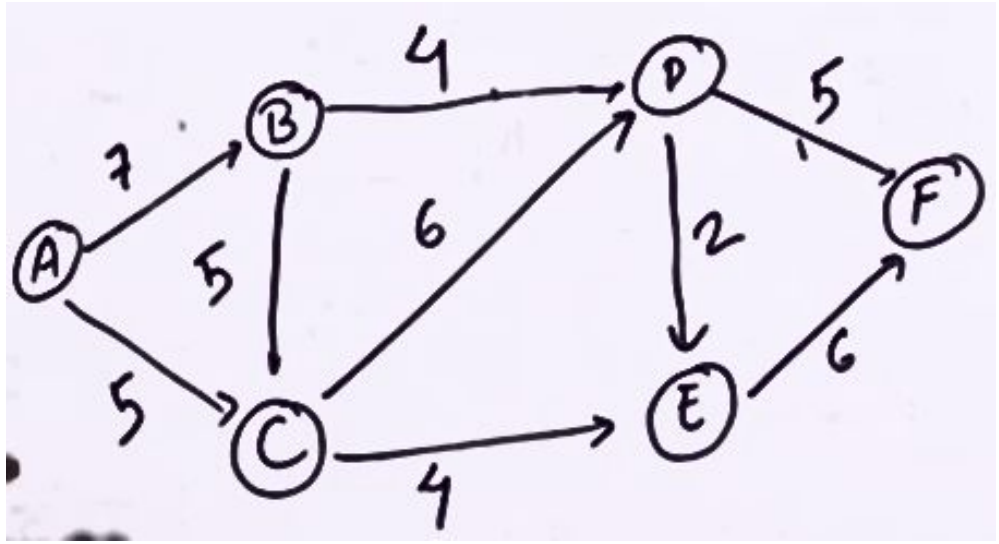
(i). Source

(ii).Sink

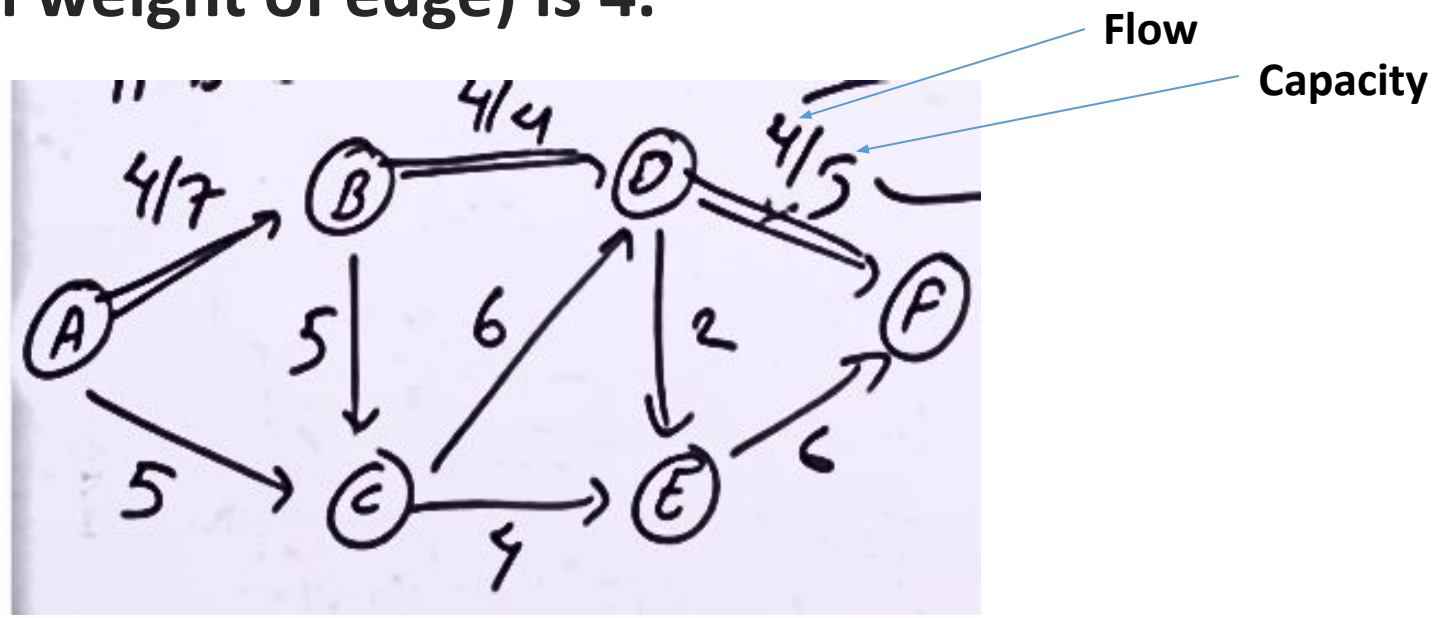
(iii). Augmented path(any path that starts with source and reach to sink).

(iv). Bottle Neck Capacity(It is the min. capacity of edge in a augmented path)

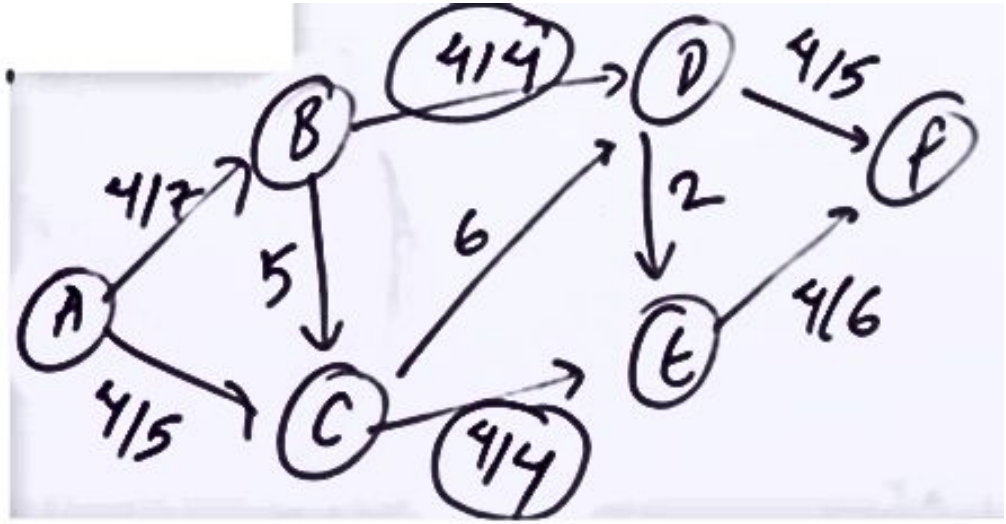
Example1: Find the maximum flow from the given network.



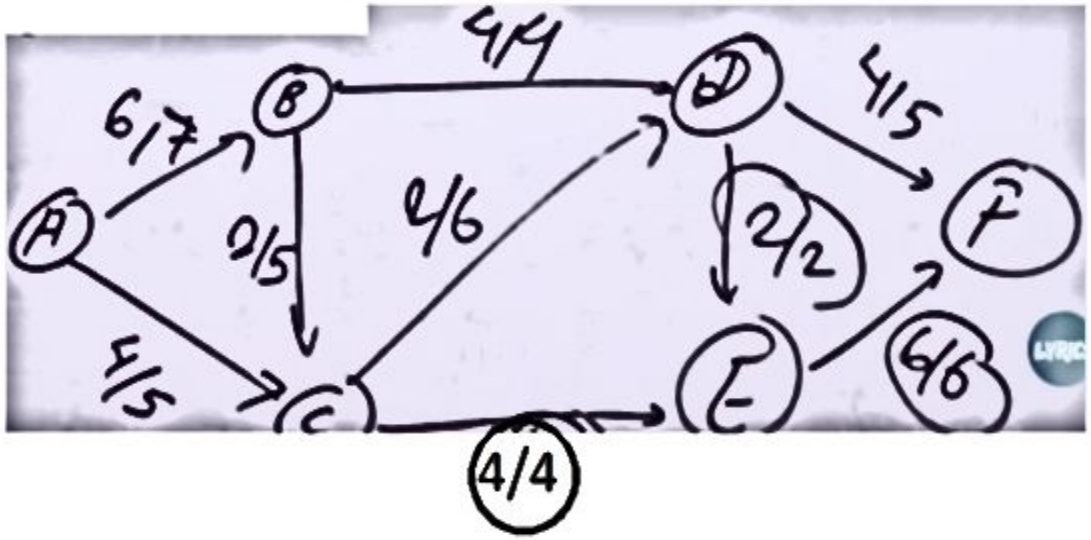
Step 1: Select any random augmented path (A□B□D□F) here ,the bottle neck capacity(min weight of edge) is 4.



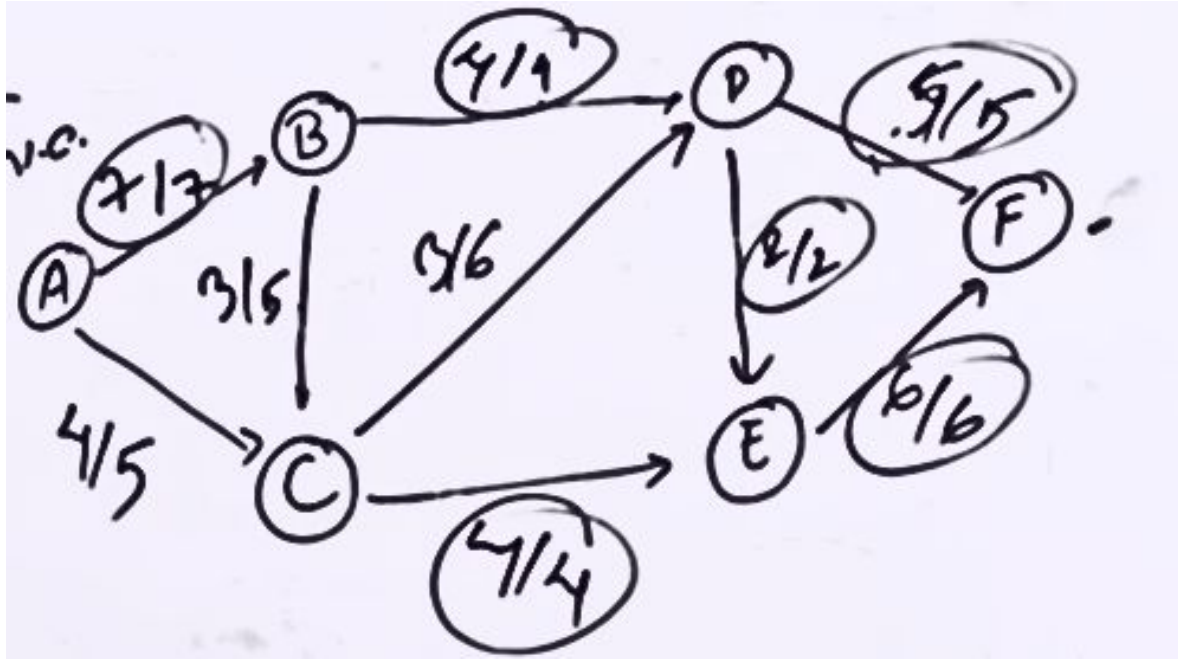
- **Step 2:** Select another random augmented path (A□C□E□F) here ,the bottle neck capacity(min weight of edge) is 4.



- **Step 3:** Select another random augmented path (A□B□C□D□E□F) here ,the bottle neck capacity(min weight of edge) is 2.



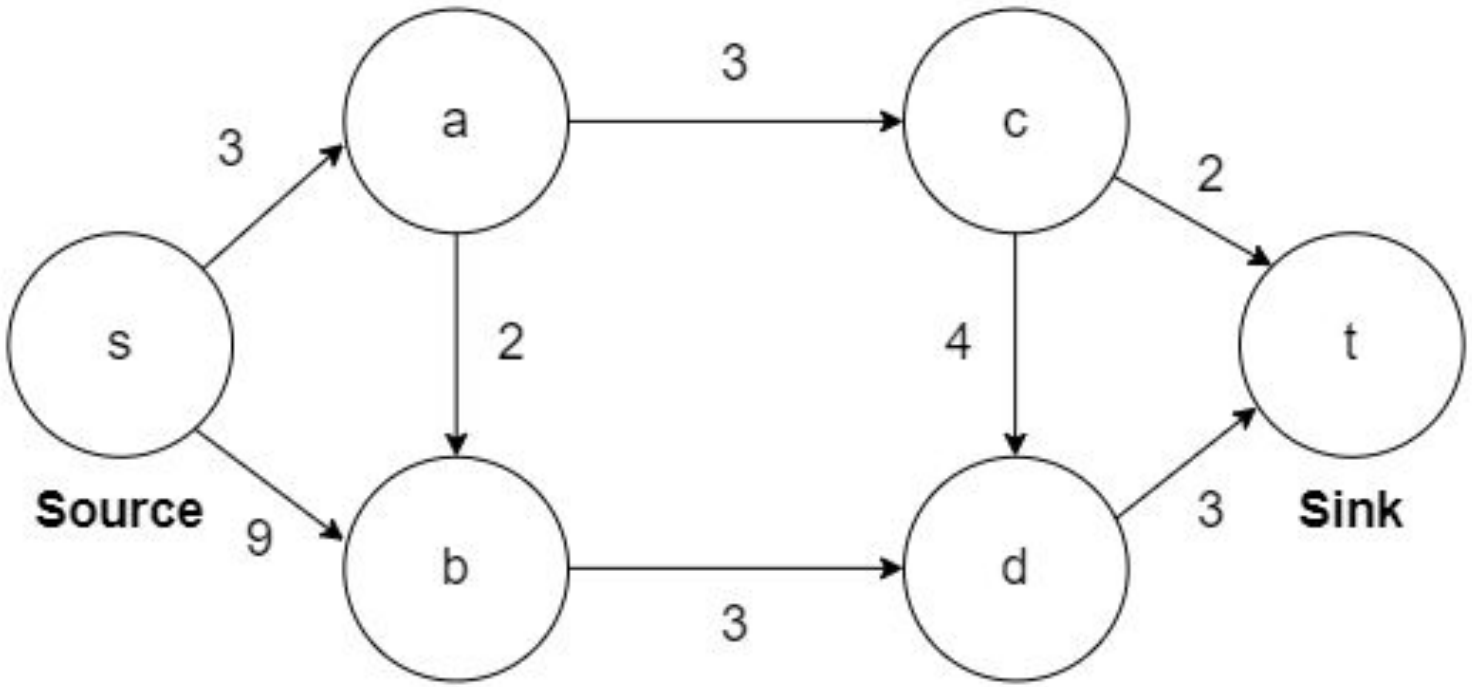
- **Step 4:** Select another random augmented path (A□B□C□D□F) here ,the bottle neck capacity(min weight of edge) is 1.



- Now ,there is no such path which is possible to travel from source to the sink at this stage.
- So, the maximum flow = All bottle neck capacity value

  = 4+4+2+1

  = 11 Ans.


a
c
s
t
b
d
3
3
2
2
4
9
3
3
Source
Sink

-The End-