UNIT-VI

***Graph Theory:** Basic Concepts of Graphs, Sub graphs, Matrix Representation of Graphs: Adjacency Matrices, Incidence Matrices, Isomorphic Graphs, Paths and Circuits, Eulerian and Hamiltonian Graphs, Multigraphs, (Problems and Theorems without proofs), Planar Graphs, Euler's Formula, Graph Colouring, Chromatic Number,(Problems and Theorems without proofs) Trees, Binary Trees, Decision Trees, Spanning Trees: Properties, Algorithms for Spanning trees and Minimum Spanning Trees*

GRAPH THEORY

Basic concepts of graphs:

Graph:- A graph **G** consists of a pair **(V,E)** where **V** is a non-empty finite set whose elements are called ***vertices*** (or nodes or points) and **E** is another set whose elements are called ***edges*** (or lines) such that each edge eϵE is associated with ordered or unordered pair of elements of V, that is there is a mapping from the set of edge E to the set of ordered or unordered pair of elements of V.

The graph G with vertices V and edge E is written as G = (V,E) or G(V,E).

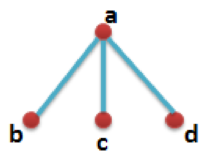
(or)

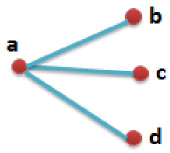
A graph G = (V,E) consists of two objects V and E such that V = $\{v_1,v_2,........\}$ called vertices and E = $\{e_1,e_2,......\}$ called edges and each edge e_k is associated with an unordered pair (v_i,v_j) of vertices.

- Edge e is said to connect u and v then u,v are called as *endpoints* of e.
- The edge e that joins the nodes u and v is said to be *incident* on each of its end points u & v.
- If two distinct edges e_1 and e_2 are incident with a common point then they are called *adjacent edges*.
- Any two vertices connected by an edge in a graph are called as *adjacent vertices*.
- Vertex that is not adjacent to any other vertex is called as *isolated vertex*.

Eg-1:- ***V = {a,b,c,d} and E = {(a,b) (a,c) (a,d)} draw a graph for this information***

Sol: *G = (V,E) = (4,3)*

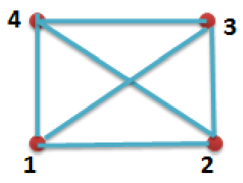




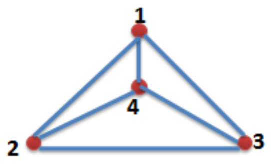
In the graph a and b are adjacent, b and c are non-adjacent.

Eg-2:- ***let V = {1,2,3,4} and E = {(1,2) (1,3) (1,4) (2,3) (2,4) (3,4)} draw a graph.***

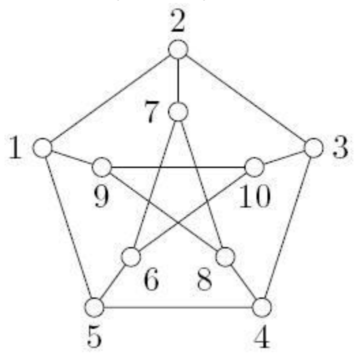
Sol: ***G = (V,E) = (4,6)***



In the graph edge (1,3) & (2,4) intersect, however their intersection is not a vertex of the graph.



Eg-3:- Consider (10, 15)



This graph is called as Petersen graph.

V = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
E = {(1,2) (2,3) (3,4) (4,5) (5,1) (1,9) (2,7) (3,10) (4,8) (5,6) (6,7) (6,10) (7,8) (8,9) (9,10)}

Types of Graphs:

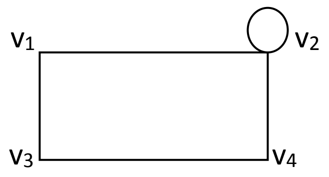
Null graph:- A graph in which number of edge is zero is called as null graph

Eg: .v_1 .v_2

.v_3 .v_4

Self loop:- An edge joining a vertex to itself is called as self loop. A loop may be defined as an edge (v_l,v_m) , where $v_l = v_m$.

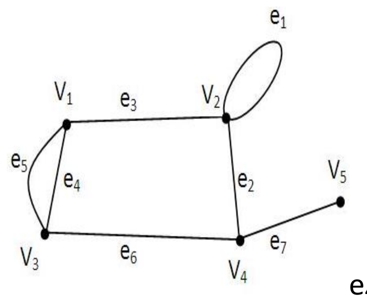
Eg:



v_2 is a self loop.

Parallel edges:- If there is more than one edge between two vertices they are called parallel edges.

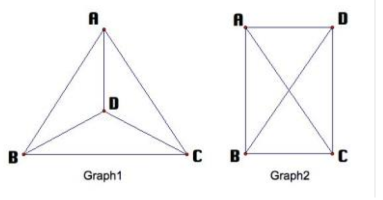
Eg:



e_4 and e_5 are parallel edges

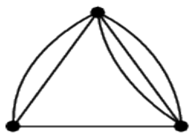
Simple Graph:- A graph which contains neither self-loops nor parallel edges is called a simple graph.

Eg:



Multi graph:- A graph which contains parallel edges is called a multi graph.

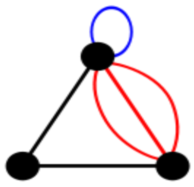
Eg:



multigraph

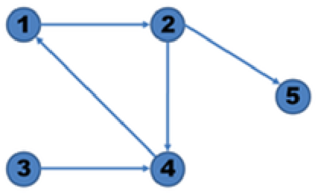
Pseudograph:- A graph in which loops and multiple edges are allowed is called a pseudo graph.

Eg:



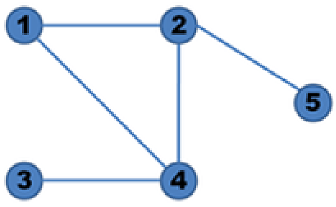
Directed graph:- A directed graph or digraph G consists of a set V of vertices and a set E of edges such that each edge e є E is associated with an ordered pair of vertices. In other words, if each edge of the graph G has direction then the graph is called directed graph.

Eg:



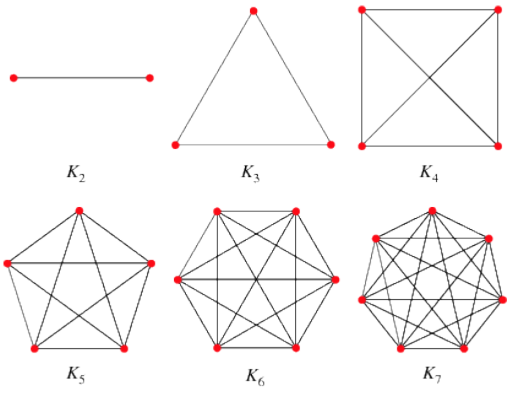
Undirected graph:- An undirected graph G consists of a set V of vertices and the set E of edges such that each edge e є E is associated with an unordered pair of vertices.

Eg:



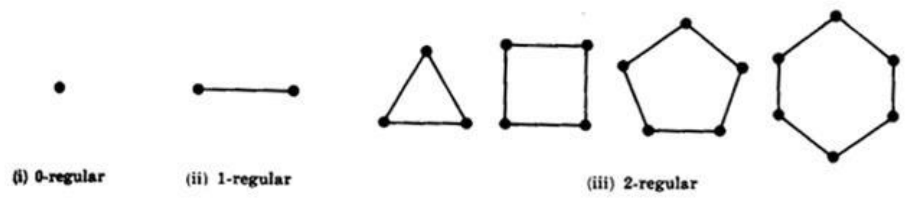
Complete graph:- A simple graph G is said to be complete if every vertex in G is connected with every other vertex if G. A complete graph is usually denoted by K_n .

Eg:



Regular graph:- A graph in which all vertices are of equal degree is called a regular graph. If the degree of each vertex is r, then the graph is called a regular graph of degree r.

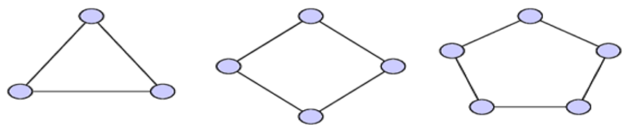
Eg:



Note:- Every null graph is a regular graph of degree zero.

Cycle Graph:- A cycle C_n , $n \geq 3$ consists of n vertice $v_1, v_2, \ldots v_n$ and edges $\{v_1,v_2\},\{v_2,v_3\}\ldots\ldots\{v_{n-1},v_n\}$ and $\{v_n,v_1\}$.

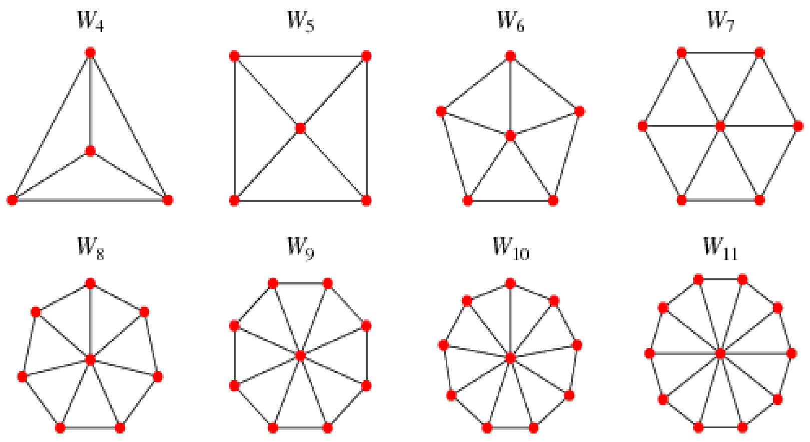
Eg:



C_3 C_4 C_5

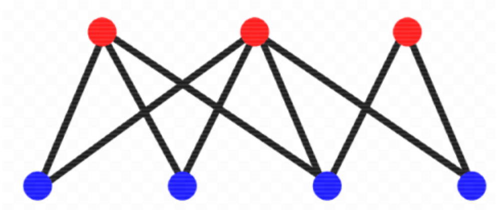
Wheels:- A wheel W_n is obtained when an additional vertex is added to the cycle C_n, for n>=3 and this new vertex is connected to each of the n vertices in C_n .

Eg:



Bipartite graphs:- A graph G = (V,E) is said to be bipartite if the vertex set V can be partitioned into two disjoint subsets V_1 and V_2 such that every edge in E connects a vertex in V_1 and a vertex V_2 so that no edge in G connects either two vertices in V_1 or two vertices in V_2. (V_1,V_2) is called a bipartition of G.

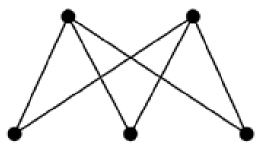
Eg:

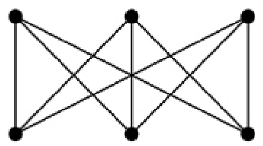


Note:- A bipartite graph can have no loop.

Complete bipartite graph:-A complete bipartite graph on m and n vertices denoted by $k_{m,n}$ is a graph whose vertex set is partitioned into sets V_1 with ***m*** vertices and V_2 with ***n*** vertices in which there is an edge between each pair of vertices v_1 and v_2 where v_1 is in V_1 and v_2 is in V_2 .

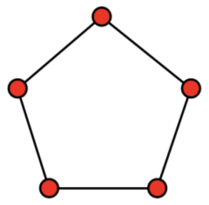
Eg:

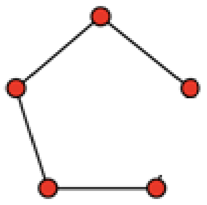

$K_{2,3}$


$K_{3,3}$

Path graph:- Path graph of order n which will be obtained by removing any one edge from cycle graph C_n and is denoted by P_n.

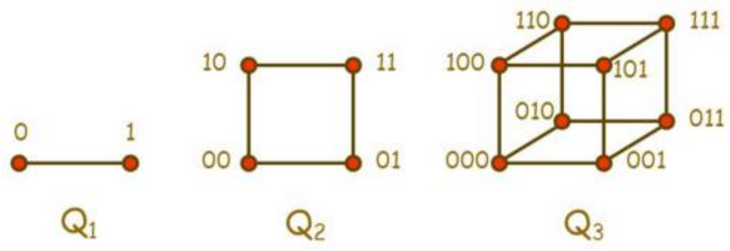
Eg:


C_5


P_5

N – cube graph:- An N-cube is a graph that has vertices representing the 2^n bit string of length n. An N-cube is denoted by Q_n.

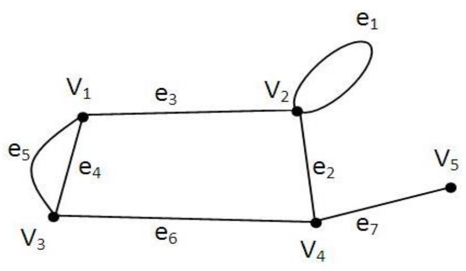
Eg:



Finite graph and Infinite graph:-A graph is finite if both its vertex set and the edge set are finite. Otherwise it is an infinite graph.

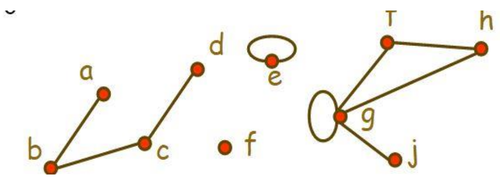
Degree of a vertex in a non-directed graph and degree sequence:-

- The degree of a vertex V of a graph G is the number of edges of G, which are incident with V.
- A vertex of degree zero is called an **isolated vertex**.
- A vertex with degree one is called a **pendant vertex**.
- A vertex of odd degree is an **odd vertex** and a vertex of even degree is an **even vertex**.



$Deg(V_1) = 3$
$Deg(V_2) = 4$
$Deg(V_3) = 3$
$Deg(V_4) = 3$
$Deg(V_5) = 1$

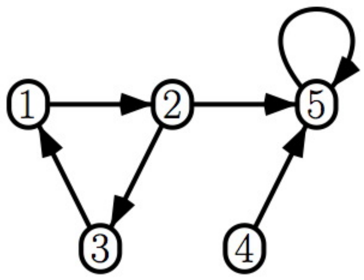
In the above graph V_5 is a pendant. The degree of V_2 is 4 because there are 2 edges incident with it and a loop which should be counted as 2.



In this example,
f is an isolated vertex.
a,d,j are pendant vertices

Degree of a vertex in a directed graph:-

- The number of edges incident to a vertex is called the ***indegree*** of the vertex and the number of edges incident from it is called its ***outdegree*** for a digraph.
- The indegree of a vertex v in a graph G is denoted by deg $G^+(v)$ and the outdegree of a vertex v in a graph G is denoted by deg $G^-(v)$.
- The degree of a vertex is determined by counting each loop incident on it twice and each other edge once.
- The minimum of all the degree of the vertices of a graph G is denoted by $\delta(G)$ and the maximum of all the degree of the vertices of G is denoted by $\Delta(G)$.

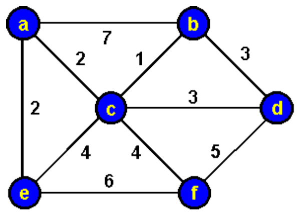


$Deg^+(1) = 1$	$Deg^-(1) = 1$
$Deg^+(2) = 1$	$Deg^-(2) = 2$
$Deg^+(3) = 1$	$Deg^-(3) = 1$
$Deg^+(4) = 0$	$Deg^-(4) = 1$
$Deg^+(5) = 3$	$Deg^-(5) = 1$

$Deg(1) = Deg^+(1) + Deg^-(1) = 1+1=2$
$Deg(2) = Deg^+(2) + Deg^-(2) = 1+2=3$
$Deg(3) = Deg^+(3) + Deg^-(3) = 1+1=2$
$Deg(4) = Deg^+(4) + Deg^-(4) = 0+1=1$
$Deg(5) = Deg^+(5) + Deg^-(5) = 3+1=4$

$\delta(G)=1$, $\Delta(G)=4$

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



Theorem-1: The sum of all vertex degrees is equal to twice the number of edges. (or) The sum of the degrees of the vertices of G is even.

$$\boxed{\sum_{i=1}^{n} d_i = 2|E|}$$

- Note:- Any graph has even number of odd degree vertices.

Theorem-2: (The handshaking theorem):- If G = (V, E) be an undirected graph with e edges then

$$\boxed{\sum_{v\in V} \deg G(v) = 2e}$$ where e is the number of edges.

Theorem-3:- If G = (V,E) be a directed graph with e edges then

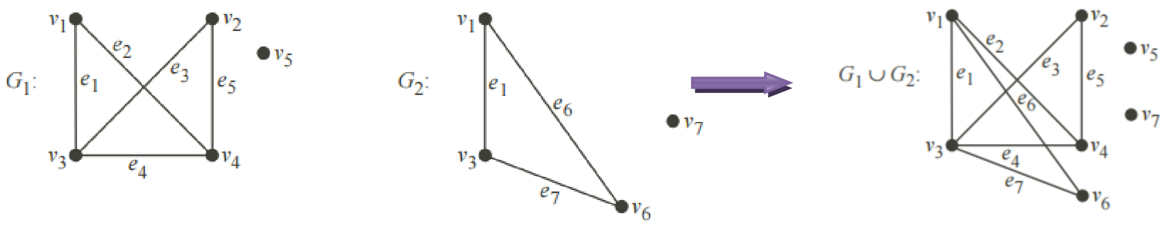
$$\boxed{\sum_{v\in V} \deg {}^{+}G(v) = \sum_{v\in V} \deg {}^{-}G(v) = e}$$

Graph Operations:-

Union of two graphs:- Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be two graphs. The union of G_1 and G_2 will be a graph (V,E) such that

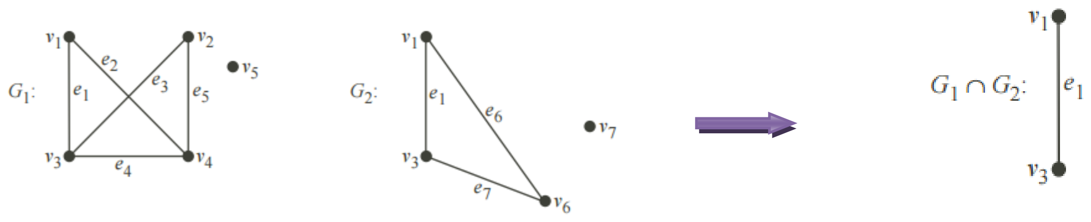
$$V = V_1 \cup V_2$$
$$E = E_1 \cup E_2$$

Eg:



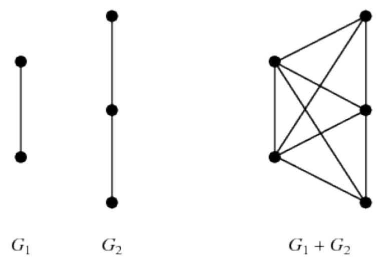
Intersection of two graphs:- Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be two graphs with atleast one vertex in common. The intersection of G_1 and G_2 will be a graph (V,E) such that $V = V_1 \cap V_2$, $E = E_1 \cap E_2$.

Eg:



Sum of two graphs:- Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be two graphs such that $V_1 \cap V_2 = \phi$. The sum G_1+G_2 is defined as the graph whose vertex set is V_1+V_2 and the edge set consists of those edges which are in G_1 and in G_2 and the edges obtained by joining each vertex of G_1 to that of G_2 .

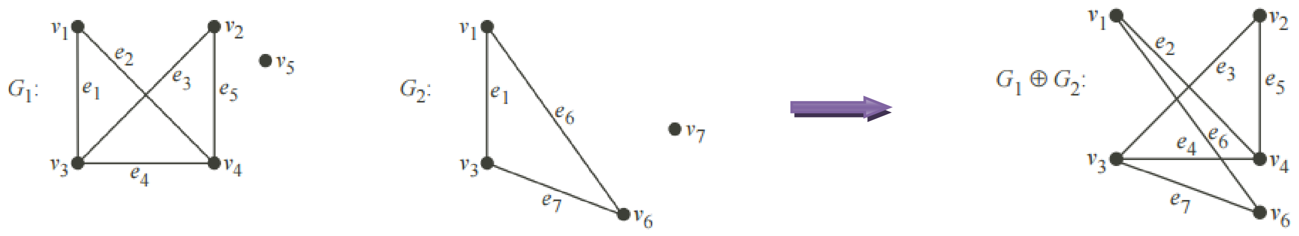
Eg:



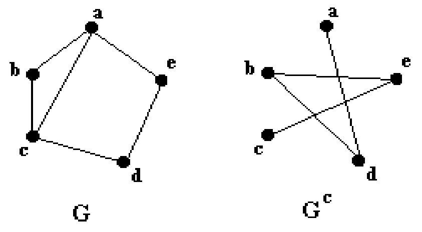
Ring Sum:- Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be two graphs. The ring sum $G_1 \oplus G_2$ is a graph G = (V,E) such that

$$V = V_1 \cup V_2$$
$$E = (E_1 \cup E_2) - (E_1 \cap E_2)$$

Eg:



Complement of a graph:- The complement $\overline{G}$ or G^C of a graph G = (V,E) is the graph with the vertex set V such that two vertices are adjacent in $\overline{G}$ if an only if these vertices are non-adjacent in G i.e, G is a graph with $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{(x,y)/(x,y) \notin E(G)\}$.



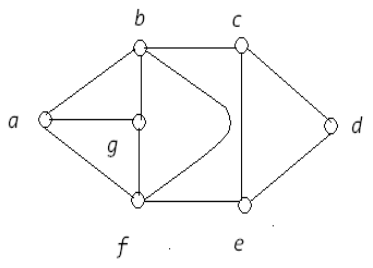
Note: Suppose G is a graph with n-vertices and m-edges. Then the number of edges in its complement is given by $\boxed{\overline{G} = \frac{n(n-1)}{2} - m}$.

For the above graph G, n=5 and m= 6. So its complement $\overline{G}$ have 5(4)/2 – 6 = 10 – 6 = 4 edges.

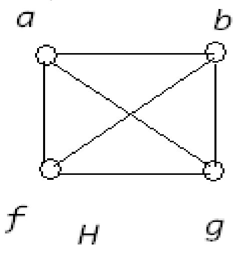
Subgraphs:

- If G and H are two graphs with vertex sets V(H) and V(G) and edge sets E(H) and E(G) respectively such that V(H)$\subseteq$V(G) and E(H)$\subseteq$E(G) , then we call H as a sub graph of G.
- If V(H)$\subset$ V(G) and E(H)$\subset$ E(G) then H is a ***proper sub graph*** of G and
- If V(H)= V(G) then we say that H is ***spanning sub graph*** of G.
- A spanning sub graph need contain all the edges in G.

  If H is a sub graph of G, then
  - ➢ All the vertices of H are in G.
  - ➢ All the edges of H are in G and
  - ➢ Each edge of H has the same end points in H as in G.

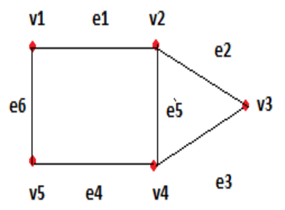


G

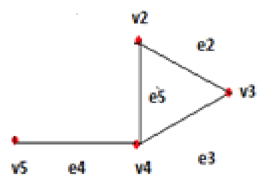


Vertex deleted subgraph:- Let G=(V,E) be a graph. Let $v_i \in$ V. The subgraph of G obtained by removing the vertex v_i and all the edges incident with v_i is called a vertex deleted subgraph and is denoted by G- v_i.

Eg:



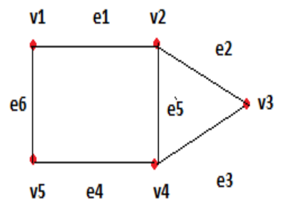
G



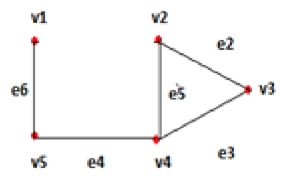
G- v_1

Edge deleted subgraph:- Let $G=(V,E)$ be a graph. Let $e_j \in E$. Then $G- e_j =(V,E-\{ e_j\})$ is called the edge deleted subgraph of G obtained by the removal of edge e_j.

Eg:



G



G- e_1

Isomorphic Graphs:- Let $G_1=(V(G_1),E(G_1))$ and $G_2=(V(G_2,E(G_2))$ be two graphs. A function $f:G_1 \rightarrow G_2$ is called an isomorphism if

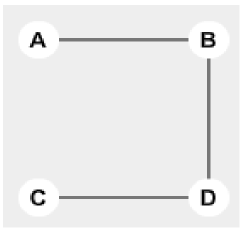
i. f is one one
ii. f is onto
iii. $(x,y) \in E(G_1)$ if and only if $f(x),f(y) \in E(G_2)$ i.e two vertices x and y are adjacent in G_1 iff $f(x),f(y)$ are adjacent in G_2. If the graph G_1 is isomorphic to G_2 then we write $G_1 \cong G_2$.

Properties of isomorphism:

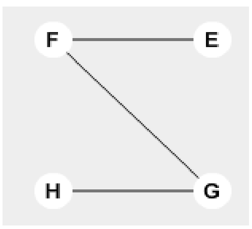
If two graphs G_1 and G_2 are isomorphic then

1. No of vertices of G_1= No of vertices of G_2
   $$|V(G_1)|=|(V(G_2)|$$
2. No of edges of G_1= No of edges of G_2
   $$|E(G_1)|=|(E(G_2)|$$
3. Degree sequences(ascending order of degrees of all the vertices) of G_1 and G_2 must be the same.
4. If (u,u) is a loop in G_1 then (f(u),f(u)) must be a loop in G_2.
5. If v_0-v_1-v_2……v_{n-1}-v_n-v_0 is cycle of length n in G_1 then $f(v_0)$-$f(v_1)$-$f(v_2)$……$f(v_{n-1})$ - $f(v_n)$ -$f(v_0)$ must be cycle of length n in G_2.
6. If two graphs are isomorphic then their adjacency matrices are same.

Eg-1: Determine whether the following graphs are isomorphic or not



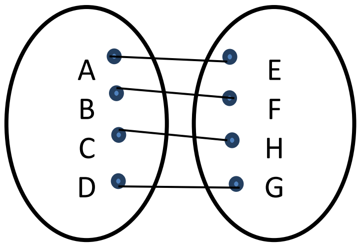
G_1



G_2

Sol:

1. Number of vertices of G_1 and G_2 are equal
   $|V(G_1)| = |V(G_2)| \quad \Rightarrow \quad 4 = 4$
2. Number of edges of G_1 and G_2 are equal
   $|E(G_1)| = |E(G_2)| \quad \Rightarrow \quad 3 = 3$
3. Mapping (Function from G_1 to G_2)
   $f(A) = E$, $f(B) = F$, $f(C) = H$, $f(D) = G$
   Clearly the given function is one-to-one and onto

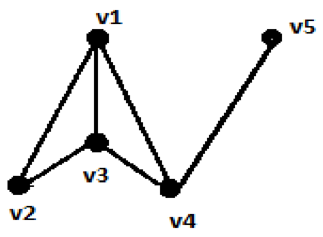


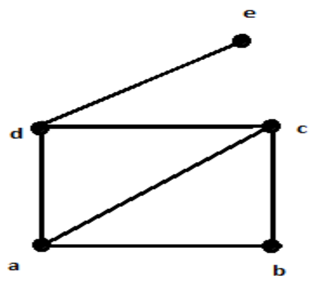
4. Degree sequence of G_1 = Degree sequence of G_2
   $(1, 1, 2, 2) = (1, 1, 2, 2)$
5. There are no loops in G_1 and G_2.
6. There are no cycles in G_1 and G_2.
7. Adjacency matrices of G_1 and G_2 are equal

$$\begin{bmatrix} & A & B & C & D \\ A & 0 & 1 & 0 & 0 \\ B & 1 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 1 \\ D & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} & E & F & H & G \\ E & 0 & 1 & 0 & 0 \\ F & 1 & 0 & 1 & 0 \\ H & 0 & 1 & 0 & 1 \\ G & 0 & 0 & 1 & 0 \end{bmatrix}$$

$\therefore$ The given graphs G_1 and G_2 are isomorphic.

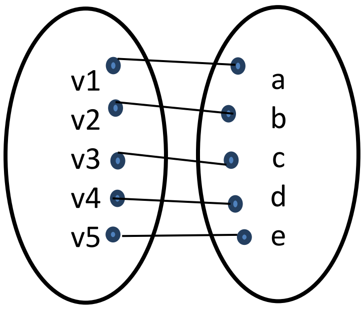
Eg-2: Determine whether the following graphs are isomorphic or not





Sol:

1. Number of vertices of G_1 and G_2 are equal
   $|V(G_1)| = |V(G_2)| \quad \Rightarrow \quad 5 = 5$
2. Number of edges of G_1 and G_2 are equal
   $|E(G_1)| = |E(G_2)| \quad \Rightarrow \quad 6 = 6$
3. Mapping (Function from G_1 to G_2)
   $f(v_4) = d$, $f(v_5) = e$, $f(v_3) = c$, $f(v_2) = b$, $f(v_1) = a$
   Clearly the given function is one-to-one and onto

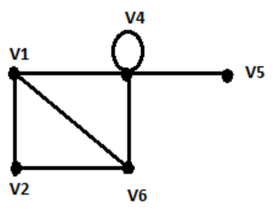


4. Degree sequence of G_1 = Degree sequence of G_2
   $(1,2,3,3,3) = (1,2,3,3,3)$
5. There are no loops in G_1 and G_2.
6. i) **v_1-v_2-v_3-v_4-v_1** is a cycle in G_1 and **$f(v_1)$-$f(v_2)$-$f(v_3)$-$f(v_4)$-$f(v_1)$** i.e **a-b-c-d-a** is a cycle in G_2 .
   ii) **v_1-v_2-v_3-v_1** is a cycle in G_1 and **$f(v_1)$-$f(v_2)$-$f(v_3)$-$f(v_1)$** i.e **a-b-c-a** is a cycle in G_2 .
   iii) **v_1-v_3-v_4-v_1** is a cycle in G_1 and **$f(v_1)$-$f(v_3)$-$f(v_4)$-$f(v_1)$** i.e **a-c-d-a** is a cycle in G_2 .
   Clearly If v_0-v_1-v_2……v_{n-1}-v_n-v_0 is cycle of length n in G_1 then $f(v_0)$-$f(v_1)$-$f(v_2)$……$f(v_{n-1})$ -$f(v_n)$ -$f(v_0)$ is a cycle of length n in G_2.
7. Adjacency matrices of G_1 and G_2 are equal

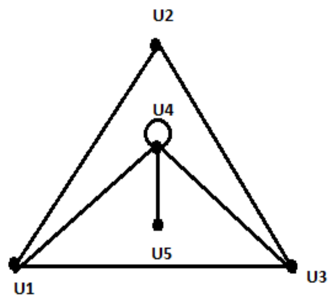
$$\begin{array}{c|ccccc} & v1 & v2 & v3 & v4 & v5 \\ \hline v1 & 0 & 1 & 1 & 1 & 0 \\ v2 & 1 & 0 & 1 & 0 & 0 \\ v3 & 1 & 1 & 0 & 1 & 0 \\ v4 & 1 & 0 & 1 & 0 & 1 \\ v5 & 0 & 0 & 0 & 1 & 0 \end{array} = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & 0 & 1 & 1 & 1 & 0 \\ b & 1 & 0 & 1 & 0 & 0 \\ c & 1 & 1 & 0 & 1 & 0 \\ d & 1 & 0 & 1 & 0 & 1 \\ e & 0 & 0 & 0 & 1 & 0 \end{array}$$

∴ The given graphs G_1 and G_2 are isomorphic.

Eg-3: Determine whether the following graphs are isomorphic or not



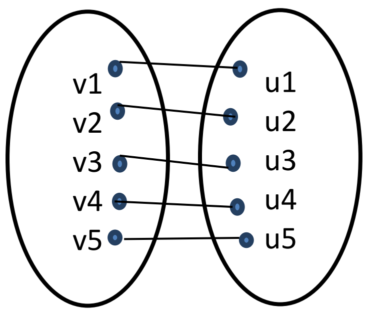
G_1



G_2

Sol:

1. Number of vertices of G_1 and G_2 are equal
   $|V(G_1)| = |V(G_2)| \Rightarrow 5 = 5$
2. Number of edges of G_1 and G_2 are equal
   $|E(G_1)| = |E(G_2)| \Rightarrow 6 = 6$
3. Mapping (Function from G_1 to G_2)
   $f(v_4) = u_4$, $f(v_5) = u_5$, $f(v_3) = u_3$, $f(v_2) = u_2$, $f(v_1) = u_1$
   Clearly the given function is one-to-one and onto



4. Degree sequence of G_1 = Degree sequence of G_2
   $(1,2,3,3,5) = (1,2,3,3,5)$
5. v4 is a loop in G1 and f(v4) i.e u4 is a loop in G2.Clearly if (u,u) is a loop in G1 then (f(u),f(u)) is a loop in G2.

6. i) **v_1-v_2-v_3-v_4-v_1** is a cycle in G_1 and **$f(v_1)$-$f(v_2)$-$f(v_3)$-$f(v_4)$-$f(v_1)$** i.e **u_1-u_2-u_3-u_4-u_1** is a cycle in G_2 .
   ii) **v_1-v_4-v_3-v_2-v_1** is a cycle in G_1 and **$f(v_1)$-$f(v_4)$-$f(v_3)$-$f(v_2)$-$f(v_1)$** i.e **u_1-u_4-u_3-u_2-u_1** is a cycle in G_2 .
   Clearly If v_0-v_1-v_2……v_{n-1}-v_n-v_0 is cycle of length n in G_1 then $f(v_0)$-$f(v_1)$-$f(v_2)$……$f(v_{n-1})$ -$f(v_n)$ -$f(v_0)$ is a cycle of length n in G_2.
7. Adjacency matrices of G_1 and G_2 are equal

$$\begin{array}{c|ccccc} & v1 & v2 & v3 & v4 & v5 \\ \hline v1 & 0 & 1 & 1 & 1 & 0 \\ v2 & 1 & 0 & 1 & 0 & 0 \\ v3 & 1 & 1 & 0 & 1 & 0 \\ v4 & 1 & 0 & 1 & 1 & 1 \\ v5 & 0 & 0 & 0 & 1 & 0 \end{array} = \begin{array}{c|ccccc} & u1 & u2 & u3 & u4 & u5 \\ \hline u1 & 0 & 1 & 1 & 1 & 0 \\ u2 & 1 & 0 & 1 & 0 & 0 \\ u3 & 1 & 1 & 0 & 1 & 0 \\ u4 & 1 & 0 & 1 & 1 & 1 \\ u5 & 0 & 0 & 0 & 1 & 0 \end{array}$$

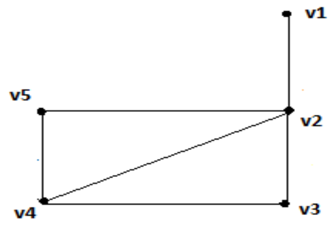
∴ The given graphs G_1 and G_2 are isomorphic.

Paths and Circuits:

Walk:

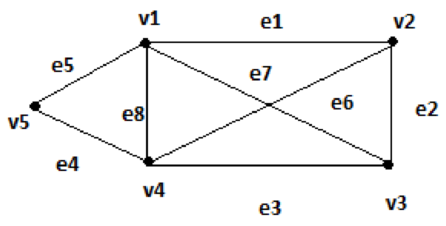
- A walk of a graph G is defined as an alternating sequence of vertices and edges $v_0,e_1,v_1,e_2,v_2,\ldots\ldots v_{n-1},e_n,v_n$ beginning and ending with vertices such that each line e_i is incident with v_{i-1} and v_i.
- A walk joining v_0 and v_n is called v_0-v_n walk. Here v_0 is called the ***initial vertex*** and v_n is called the ***terminal vertex*** of the walk
- The number of edges in the walk is known as the ***length*** of the walk.
- If the length of the walk is zero, then the walk has no edges and it contains only a single vertex. Such a walk is called a ***trivial walk***.

Trail: A walk is called a trail if its edges are distinct.
Path: A walk is called a path if all its vertices are distinct.
Note: Every path is a trail, but every trail need not be a path.
Closed path: A closed path is a path that starts and ends at the same point.
Circuits: A circuit or cycle is defined as a closed path of non zero length that does not contain a repeated edge.
Eg-1:



i) v1-v2-v3-v4-v5 is a walk of length 4(no of edges in the walk v1-v2,v2-v3,v3-v4,v4-v5)
ii) v1-v2-v4-v3-v2-v5 is a trail(edges are distinct) but not a path(because the vertex v2 is repeated)
iii) v1-v2-v4-v5 is a path and also a trail

Eg-2:

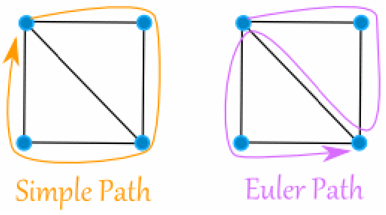


i) v1-e1-v2-e6-v4-e3-v3-e2-v2 is a walk and a trail but not a path (because v2 is repeated) and circuit.
ii) v1-e1-v2-e2-v3-e3-v4-e4-v5 is a walk, trail and a path but not a circuit
iii) v2-e2-v3-e3-v4-e4-v5-e5-v1-e1-v2 is a walk, trail, path and a circuit

Eulers (or) Eulerian Graph:

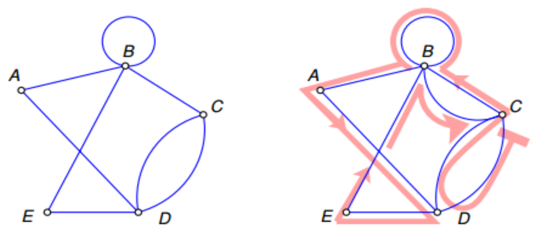
1. An ***Euler's path*** in a multigraph is a path that travels all edges exactly once and vertices can be travelled at least once.

2. In any multigraph we can get an Euler's path if there are two vertices of odd degree and remaining vertices degree is even.
3. The Euler's path will always start at odd degree vertex and end at other odd degree vertex.



Euler's circuit/cycle:

If the starting and ending vertices of Euler's path are same then such a path is called Euler's circuit.

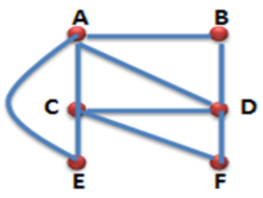


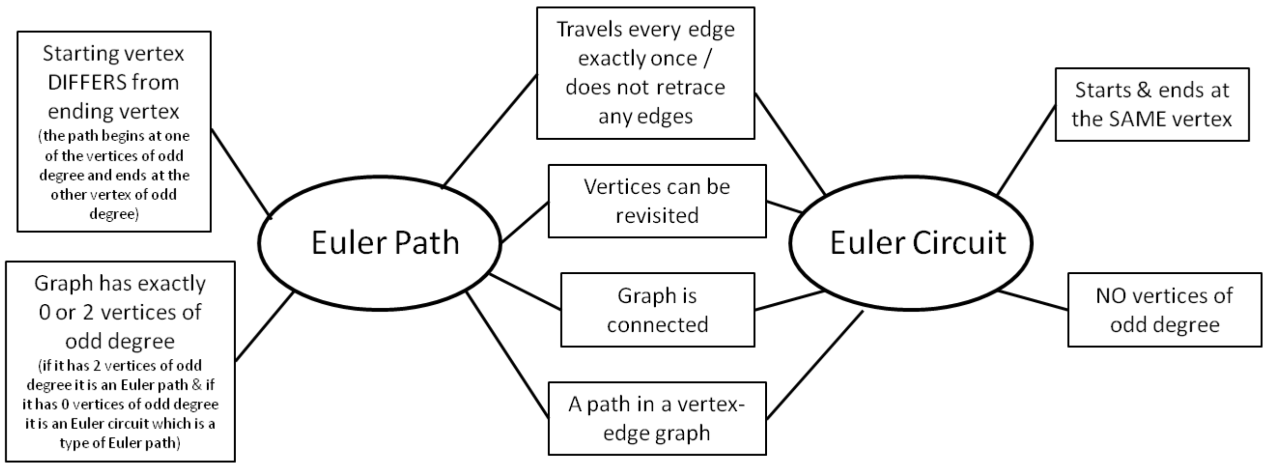
An Euler circuit: CDCBBADEBC

Euler's graph:

A graph having an Euler's circuit is called an Euler's graph.

Eg:



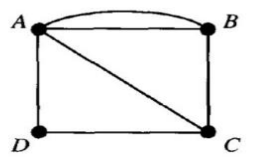

Starting vertex DIFFERS from ending vertex (the path begins at one of the vertices of odd degree and ends at the other vertex of odd degree)
Graph has exactly 0 or 2 vertices of odd degree (if it has 2 vertices of odd degree it is an Euler path & if it has 0 vertices of odd degree it is an Euler circuit which is a type of Euler path)
Euler Path
Travels every edge exactly once / does not retrace any edges
Vertices can be revisited
Graph is connected
A path in a vertex-edge graph
Euler Circuit
Starts & ends at the SAME vertex
NO vertices of odd degree

Theorem 1: If G is a graph in which the degree of every vertex is atleast two then G contains a cycle.
Theorem 2: A non-empty connected graph G is Eulerian if and only if its vertices are all of even degree.

Hamiltonian graph:

Hamiltonian circuit: A circuit in a graph G is called a Hamiltonian graph if it contains every vertex of G exactly once, except for the starting and ending vertex that appears twice.

Eg:

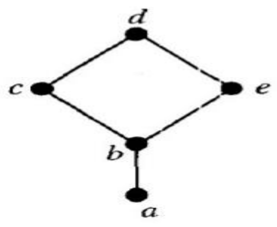


Hamiltonian circuit: A, D, C, B, A

Hamiltonian graph: A graph is said to be a Hamiltonian graph if it contains Hamiltonian cycle i.e a graph G is Hamiltonian if there exists a cycle containing every vertex of G.

Hamiltonian path:A path in a graph G is called a Hamiltonian graph if it contains every vertex of G where the end points may be distinct.

Eg:



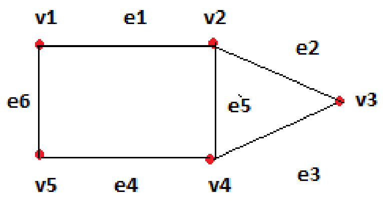
Hamiltonian path: a, b, c, d, e

Note: Hamiltonian path in G may be obtained by deleting an edge from a Hamiltonian cycle/

Note: Every graph which has a Hamiltonian cycle or circuit contains a Hamiltonian path but a graph containing Hamiltonian path may not have a Hamiltonian cycle.

Theorem 1: (Dirac's theorem) Every graph G with n ≥ 3 vertices.If deg(v) ≥ n/2 for all vertices of G, then G is ***Hamiltonian.***

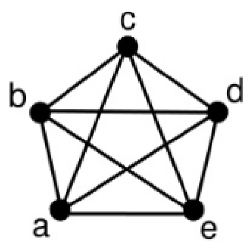
Eg-1:



The given graph has n=5 which is ≥ 3 vertices and n/2 =5/3 =2.deg(v1) =2, deg(v2) =3, deg(v3) =2, deg(v4) =3 and deg(v5) =2.

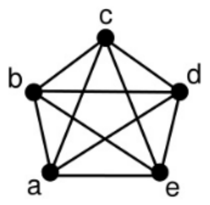
∴ deg(v) ≥ n/2 for all vertices of G. So the given graph is Hamiltonian.

Theorem 2: If the graph G has no loops or parallel edges, if V|G|= n ≥ 3 and if deg(v) ≥ n/2 for each vertex v of G, then G is Hamiltonian.



Given graph has no loops, no parallel edges,V (G) = 5 which is ≥ 3, n/2= 5/2 =2
deg (a) = 4, deg (b) = 4, deg(c) = 4,
deg (d) = 4, deg (e) = 4 and deg (v) ≥ n/2
i.e. 4 ≥ 2 for each vertex v of G.
∴ The given graph is Hamiltonian and circuit is a, b, c, d, e

Theorem 3: Let G be a graph with n vertices with no loops or parallel edges. Then, G has atleast ½ (n-1) (n-2)+2 edges in Hamiltonian.

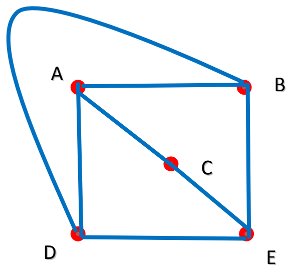


Given graph has 5 vertices & it has no loops and parallel edges.

$$= \frac{(n-1)(n-2)}{2} + 2 = \frac{(5-1)(5-2)}{2} + 2 \quad = \frac{(4)(3)}{2} + 2$$

= 6+2 =8.Hence the graph has atleast 8 edges (Minimum 8 edges and they may be exceed 8, in this case 10) .So the given graph is a Hamiltonian graph.

Theorem 4: If the graph G has no loops or parallel edges, if V|G|= n $\geq$ 3. If deg(u) + deg(v) $\geq$ n for each two vertices u and v not connected by an edge in G is Hamiltonian.



Given graph has n=5 which is $\geq$ 3 vertices. First find all the possible vertices u and v not connected by an edge in G.

S.No	u	v	deg(u)	deg(v)	deg(u)+ deg(v)
1	A	E	3	3	6
2	B	C	3	2	5
3	C	D	2	3	5

Clearly it is evident from the table that deg(u) + deg(v) $\geq$ n for each two vertices u and v not connected by an edge in G. So the given graph is Hamiltonian.

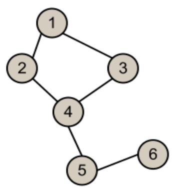
Representation of graphs:-

Matrix representation of graphs: A diagrammatic representation of a graph is very useful for visual study. Any graph can be represented by a matrix. A matrix is a very effective and convenient way of representing a graph. There are two types of matrix representation.

1. Adjacency matrix
2. Incident matrix

1. ***Adjacency matrix:*** Representation of different types of graphs as adjacency matrices is described as follows.
   a. *Representation of undirected graph:*
   Let G be a graph with n vertices, where n>0 and no parallel edges. The adjacency matrix of G denoted by A_G is an nxn matrix, A=[a_{ij}], whose elements are defined as follows:
   a_{ij}=1, if there is an edge between ith and jth vertices
   =0, if there is no edge between them.

Undirected Graph & Adjacency Matrix



Undirected Graph

$$\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 1 & 0 & 0 \\
3 & 1 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$

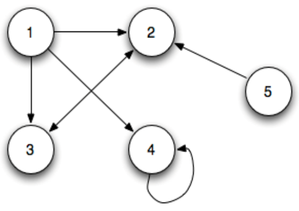
Adjacency Matrix

b. *Representation of directed graph:*

The adjacency matrix of a digraph D with n vertices is an nxn matrix $A_G=(a_{ij})$ in which

$a_{ij} = 1$, if (v_i,v_j) is in D

$= 0$, otherwise



$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 1 & 1 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 \\
5 & 0 & 1 & 0 & 0 & 0
\end{array}$$

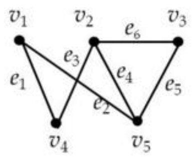
2. *Incidence matrix:*

a. *Representation of undirected graph:*

Let G=(V,E) be an undirected graph with n labeled vertices and m labeled edges. The incidence matrix $I_G=(b_{i,j})$ is the nxm matrix , where

$b_{ij} = 1$, if the jth edge e_j is incident on the ith vertex v_i

$= 0$, otherwise



$$\begin{array}{c|cccccc}
 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\
\hline
v_1 & 1 & 1 & 0 & 0 & 0 & 0 \\
v_2 & 0 & 0 & 1 & 1 & 0 & 1 \\
v_3 & 0 & 0 & 0 & 0 & 1 & 1 \\
v_4 & 1 & 0 & 1 & 0 & 0 & 0 \\
v_5 & 0 & 1 & 0 & 1 & 1 & 0
\end{array} \longleftarrow \text{edges}$$

$\uparrow$ vertices

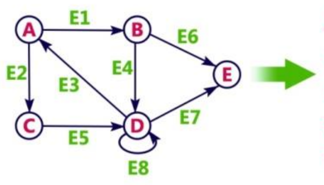
b. *Representation of directed graph:*

The incidence matrix $I_G = (b_{ij})$ of digraph D with n vertices and m edges is the nxm matrix, where

$b_{ij} = 1$, if the jth edge is incident out of the ith vertex.

$= -1$, if the jth edge is incident into the ith vertex.

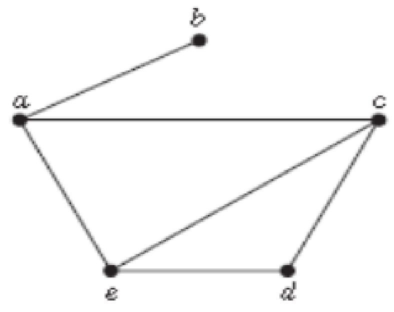
$= 0$, if jth edge in not incident on ith vertex.

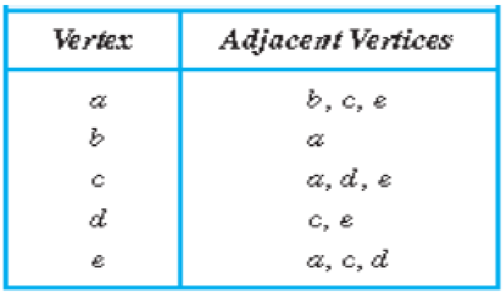


$$\begin{array}{c|cccccccc}
 & E1 & E2 & E3 & E4 & E5 & E6 & E7 & E8 \\
\hline
A & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
B & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
C & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
D & 0 & 0 & 1 & -1 & -1 & 0 & 1 & 1 \\
E & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0
\end{array}$$

Linked Representation: In this a list of vertices adjacent to each vertex is maintained. This is also called adjacency list representation.

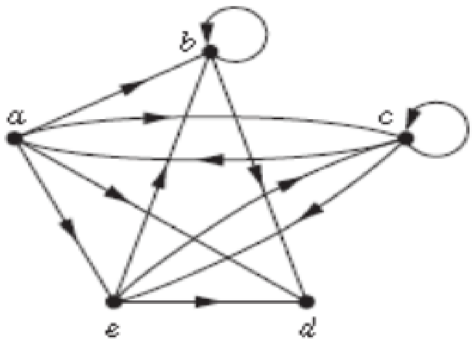
The adjacency list for simple undirected graph is shown below:

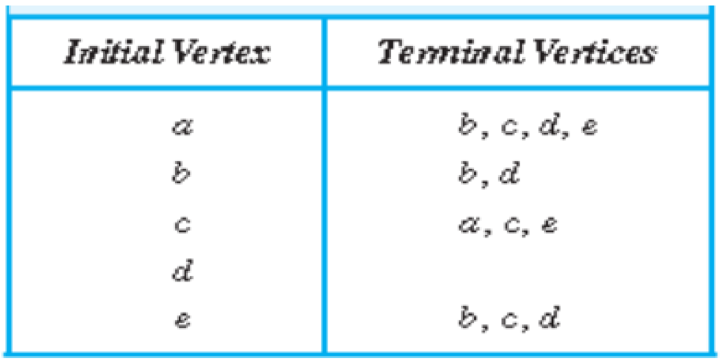




Vertex	*Adjacent Vertices*
a	*b, c, e*
b	*a*
c	*a, d, e*
d	*c, e*
e	*a, c, d*

The adjacency list for a simple directed graph is shown below:



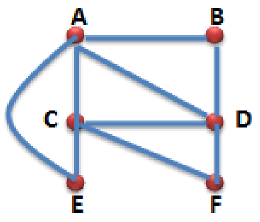


Initial Vertex	*Terminal Vertices*
a	*b, c, d, e*
b	*b, d*
c	*a, c, e*
d	
e	*b, c, d*

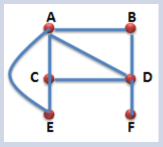
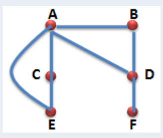
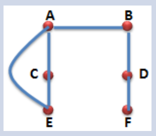
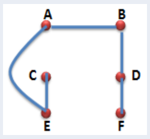
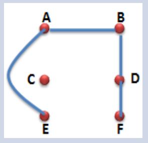
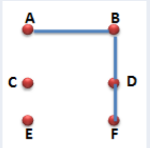
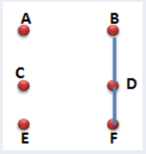
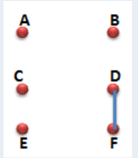
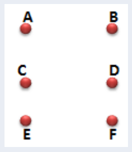
Fleury's Algorithm for finding Euler's circuit:- Let G = (V,E) be an Eulerian connected graph with each vertex of even degree. The following steps may be used to construct an Eulerian circuit.
Step 1:- Choose any arbitrary vertex v_0 from V as the starting vertex.
Step 2:- Select an edge e = (v_0,v). If there are many such edges, select the one that is not a bridge. If there are more than one edges of that type choose any of them. If e is a bridge, select only if there is no possibility.
Step 3:- Now delete that edge from the edge list and the graph.
Step 4:- Repeat step-2 and step-3 until $E=\phi$.

Note:- To find an Euler's path in a graph, Fleury's algorithm can be used with a slight modification. The choice of selection of starting vertex is limited to one of the two odd – degree vertices. If all the vertices are of even degree then no modification is required.

Eg-1: Find Euler's circuit for the following graph

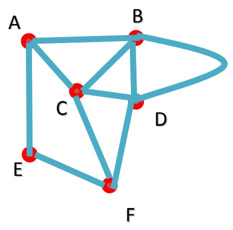


Sol:

Current Path	Next Edge	Reason | Remark	
1.F	{ F, C }	No edge from F is a bridge. So choose anyone. Let it be C i.e, {F, C}	
2. F, C	{ C, D }	No edge from C is a bridge. So choose anyone. Let it be D i.e, {C, D}	
3. F, C, D	{ D, A }	No edge from D is a bridge. So choose anyone. Let it be A i.e, {D, A}	
4.F, C, D,A	{ A,C }	From A edges are {A,B}, {A,C}, {A,E}. Since {A, B} is a bridge choose the remaining edges. Let it be {A, C}.	
5. F,C,D,A,C	{ C, E }	From C there is only one edge {C,E}. So choose it.	
6. F,C,D,A,C,E	{ E, A }	From E there is only one edge {E,A}. So choose it.	
7. F,C,D,A,C, E, A	{ A, B }	From A there is only one edge {A,B}. So choose it.	
8. F,C,D,A,C, E, A, B	{ B, D }	From B there is only one edge {B,D}. So choose it.	
9. F,C,D,A,C, E, A,B,D	{ D, F}	From D there is only one edge {D,F}. So choose it.	
10. F,C,D,A,C,E,A,B,D, F	ϕ	No more edges left	

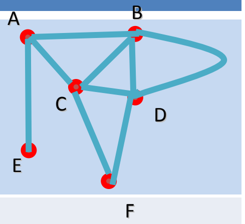
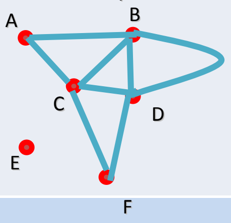
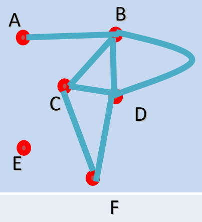
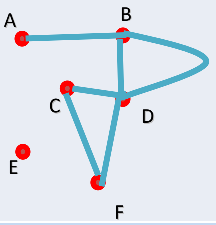
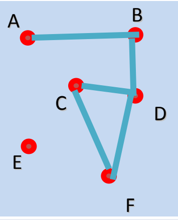
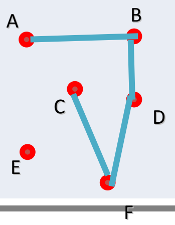
∴ Euler's circuit is ***F, C, D, A, C, E, A, B, D, F***

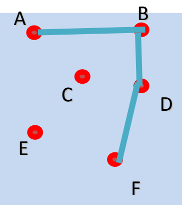
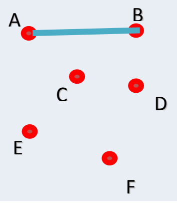
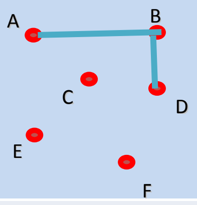
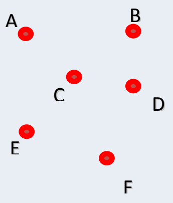
Eg-2:



Sol:
deg(A)=3,deg(B)=deg(C)=deg(D)=4,deg(E)=2 and deg(F)=3. So the given graph has 2 odd degree vertices A and F. So only path exists. So start at any odd degree vertex. Let's start at F.

Note: Euler's path starts at one degree vertex and ends at the other odd degree vertex.

Current Path	Next Edge	Reason | Remark	
1.F	{ F, E }	No edge from F is a bridge. So choose anyone. Let it be {F, E}	
2. F, E	{ E, A }	From E there is only one edge {E,A}. So choose it.	
3. F, E,A	{ A, C }	No edge from A is a bridge. So choose anyone. Let it be {A, C}	
4.F, E,A,C	{ C,B }	From C there are 3 edges {C,B},{C,D} and {C,F} out of which {C,F} is bridge. So select one from remaining two edges. Let it be {C,B}	
5. F,E,A,C,B	{ B, D }	From B there is a multiple edge {B,D} and another edge {B,A} which are not bridges. So choose anyone. Let it be {B,D}	
6. F,E,A,C,B,D	{ D, C }	From D, {D,B} is bridge. So choose from {D,C} or {D,F}.Let it be {D,C}.	

7. F,E,A,C,B,D,C	{ C, F }	From C there is only one edge {C,F}.So choose it	
8. F,E,A,C,B,D,C,F	{ F, D }	From F there is only one edge {F,D}.So choose it	
9. F,E,A,C,B,D,C,F,D	{ D, B}	From D there is only one edge {D,B}.So choose it	
10. F,E,A,C,B,D,C,F,D,B	{ B, A}	From B there is only one edge {B,A}.So choose it	
11. F,E,A,C,B,D,C,F,D,B,A	ϕ	No more edges left	

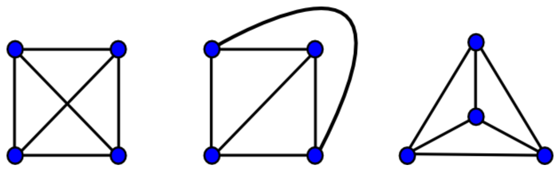
∴ Euler's path is **F,E,A,C,B,D,C,F,D,B**

Planar graphs:

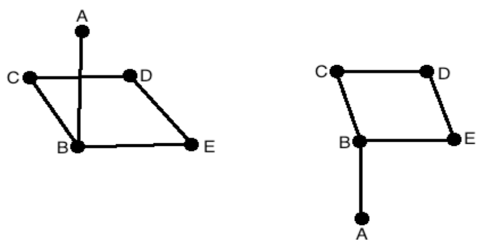
A graph G is called a planar graph if it can be drawn on a plane such that no two edges intersect except at the common vertex.

→ A graph may be planar even if it is usually drawn with crossings since it may be possible to draw it in a different way without crossings.

Eg-1:



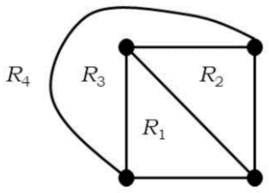
Eg-2:



Euler's theorem/ Euler's Formula:

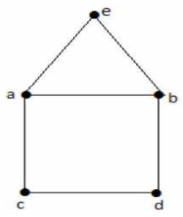
If a connected planar graph 'G' has n_v vertices, n_e edges and n_f faces or regions then $n_v - n_e + n_f = 2$.

Eg-1:



Sol: For the given graph $n_v = 4$, $n_e = 6$ and $n_f = 4$ and $n_v - n_e + n_f = 4-6+4 = 8-6 = 2$. So it is a planar graph.

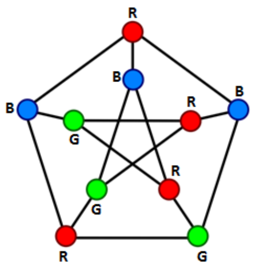
Eg-2:



Sol: : For the given graph $n_v = 5$, $n_e = 6$ and $n_f = 3$ and $n_v - n_e + n_f = 5-6+3 = 8-6 = 2$. So it is a planar graph.

Graph coloring:

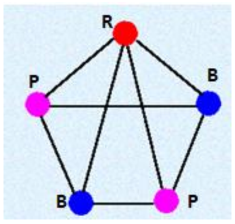
→ An assignment of colors to the vertices of a graph so that no two adjacent vertices get the same color is called coloring of the graph or simply vertex coloring.
→ The 'n' coloring of G refers to the coloring of G using n-colors.
→ If G has n-coloring then G is said to be n-colorable.



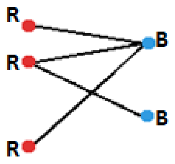
Chromatic number:

- The chromatic number of a graph G is the minimum number of colors needed to color the vertices of a graph G is denoted by $\chi(G)$.
- A graph G is n-colorable if $\chi(G) \leq$ n.

Eg's :



Chromatic number = 3



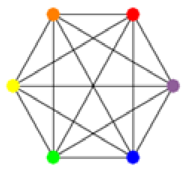
Chromatic number = 2

The chromatic number of some familiar graphs:

graph G	Chromatic number $\chi(G)$
complete graph K_n	$\chi(K_n) = n$
graph complement $\overline{K_n}$	$\chi(\overline{K_n}) = 1$
cycle C_n	$\chi(C_n) = 2$, for n – even $\chi(C_n) = 3$, for n – odd
wheel W_n	$\chi(W_n) = 4$, for n – even $\chi(W_n) = 3$, for n – odd
star S_n	$\chi(S_n) = 2$
tree T	$\chi(T) = 2$

- Theorem:- Let G be a non-trivial simple graph. Then $\chi(G)$=2 if and only if G is a bipartite graph.
- Theorem:- For any simple graph G, $\chi(G) \le \Delta(G)+1$.

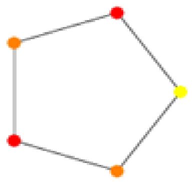
- Determine the chromatic number of complete graph K_n with n vertices.



$\boxed{\chi = 6}$

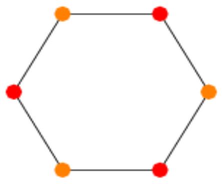
$\chi(K_n) = n$

- Determine the chromatic number of C_n if n = even and n = odd.
  Sol:- $\chi(C_n) = 2$, if n is even.
  $\chi(C_n) = 3$, if n is odd.



$\chi = 3$

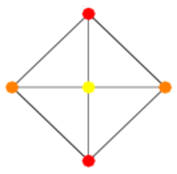
$n = 5$



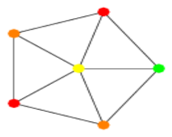
$\chi = 2$

$n = 6$

- Determine the chromatic number of wheel graph with n vertices.
  Sol:- $\chi(w_n) = 3$, if n is odd.
  $\chi(w_n) = 4$, if n is even.



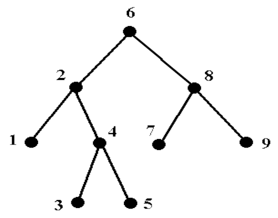
$\chi = 3$, n=5

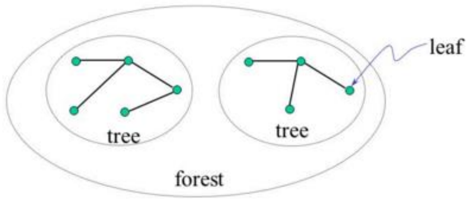


$\chi = 4$, n=6

Trees:

A graph that contains no cycles is called **acyclic graph** and a connected acyclic graph is called a **tree**. In other words, a connected graph with no cycles is known as a tree. Its edges are called branches. Connected: There is a path between any pair of nodes.

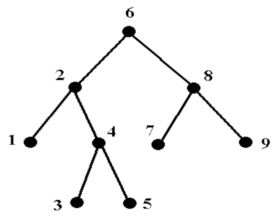




Tree properties:

i. There is only one path between every pair of vertices, in a tree T.
ii. If in a graph G, there is one and only one path between every pair of vertices then G is a tree.
iii. A tree with n vertices has (n-1) edges.
iv. For any +ve integer n, if G is a connected graph with n vertices and n-1 edges , then G is a tree.
v. In a tree with more than one vertex there are at least two vertices of degree 1.

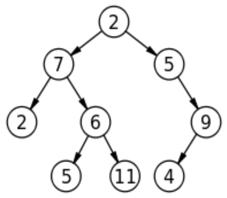
Rooted Trees:- A rooted tree is a tree in which a particular vertex is distinguished from the others and is called the **root**.
Height:- It is defined as maximum level.
Depth:- It is defined as the length of the path from the root to V.



- Root of Tree = 6
- Find all leaves vertices
  1 ,3, 5, 7, 9
- What's the level of 4 and 8 → 2 and 1
- Children of 8 are 7 and 9.
- Internal vertices are 2, 8 and 4

m-ary tree: A rooted tree is an m-ary tree if every internal vertex has **atmost** m children.

Binary Tree: A 2-ary tree is called a binary tree. A binary tree is a rooted tree in which each vertex has at most two children(means every vertex should have 0 children or I child or 2 children), designated as left child and right child. If a vertex has one child, that child is designated as either a left child or a right child, but not both. A full binary tree is a binary tree in which each vertex has exactly two children or none.



Note: If T is a full binary tree with i internal vertices, then T has i+1 terminal vertices and 2i + 1 total vertices.

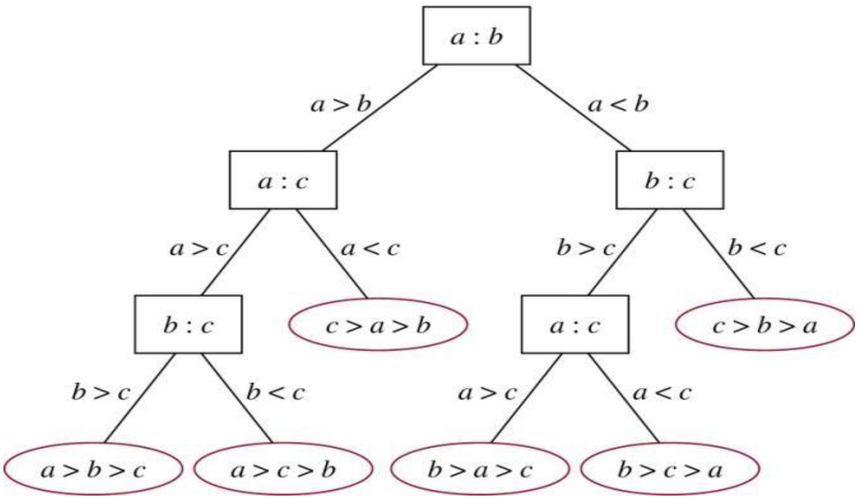
Decision Trees: Decision trees are those which are used to model problems in which a series of decisions lead to a solution.

Eg-1: Sort three distinct elements a, b, and c

Sol:

Case (i) : Let a>b. Then compare a and c.
If a<c then c>a>b.
If a >c , then compare b and c.
If b<c then a>c>b, otherwise a>b>c.
Case (ii): The case a< b can be discussed similarly.

The following tree is the decision tree for this problem.



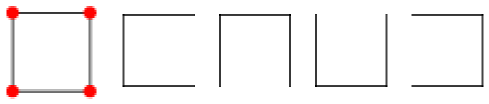
Spanning Trees:

A sub graph 'T' of a graph 'G' said to be a spanning tree if

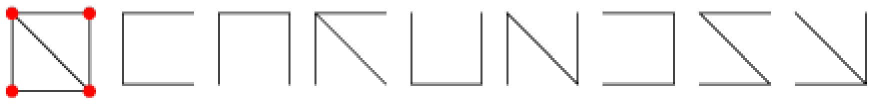
a. 'T' is a tree.
b. If 'T' includes every vertex of 'G' i.e. V(T) = V(G)

Below are the examples of some graphs and all possible spanning trees of them.

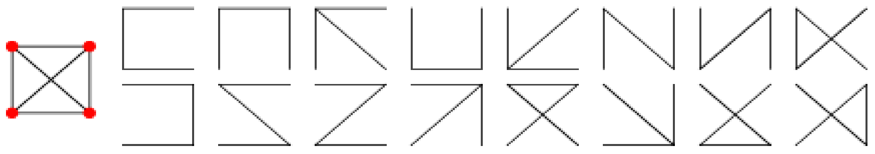
Eg-1:



Eg-2:



Eg-3:



Algorithms for spanning trees:

1. BFS
2. DFS

BFS (Breadth First Search): In this algorithm a rooted tree is constructed and the undirected graph of this rooted tree forms the spanning tree.
The idea of BFS is to visit all the vertices on a given level before going to the next level.

Procedure:

Step1: Choose a vertex arbitrarily and designated it as the root.

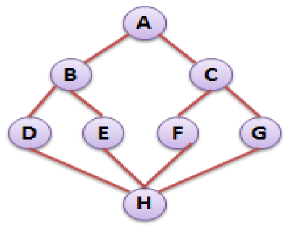
Step2: Add all the edges incident to this vertex such that the addition of vertex does not produce any cycle. The new vertices added at this stage become the vertices at level **1** in the spanning tree. Order them in as arbitrary manner.

Step3: For each vertex at level **1**, visited in order, add each edge incident to vertex to the tree as long as it does not produce any cycle.

Step4: Arbitrarily order the children of each vertex at level **1**. This produces the vertices at level **2** in the tree.

Step5: Continue with the same procedure until all the vertices in the tree have been added.
This procedure ends, since there are only a finite no of edges in the graph.
Since we have produced a tree without cycle containing every vertex of the graph, the produced tree is a spanning tree.

Eg: Find Breadth First Search (BFS) order for the given graph:



Graph

Sol:

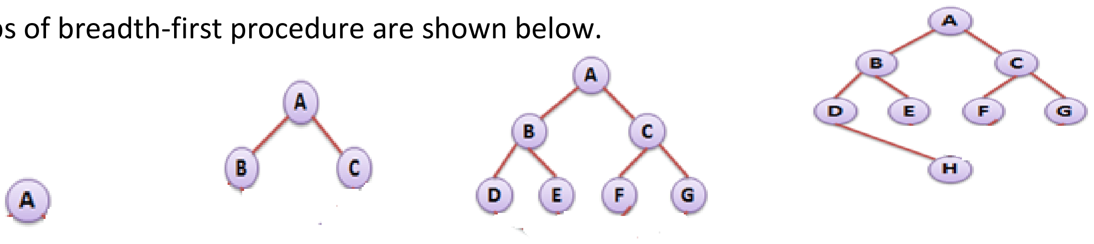
Step 1: Choose the vertex 'A' to be the root.

Step-2: Add all the edges incident to the vertex A such that the addition of edges does not produce any cycle. Hence the edges {A,B},{A,C} are added. The two vertices B and C are in level **1** in the tree.

Step-3: Add edges from these vertices at level 1 to adjacent vertices which are not already in the tree. Hence the edges {B,D}, {B,E}, {C,F} and {C,G} are added. The vertices D,E,F and G are in level **2** in the tree.

Step-4: Add edges from these vertices at level 2 to adjacent vertices which are not already in the tree. Hence the edges {D,H} is added. The vertex H is in level **3.**
in the tree.

The steps of breadth-first procedure are shown below.



(a) (b) (c) (d)

So the required path is A-B-C-D-E-F-G-H and the BFS spanning tree is fig(d) given above.

DFS (Depth First Search): DFSA is also known as backtracking.

Procedure:

Step1: Arbitrarily choose a vertex from the vertices of the graph and designate it as the root.

Step2: Form a path starting at this vertex by successively adding edges as long as possible where each new edge is incident with the last vertex in the path without producing any cycle.

Step3: If the path goes through all the vertices of the graph, then the tree consisting of this path is a spanning tree otherwise go the next step.

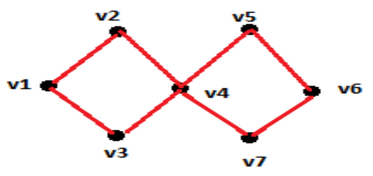
Step4: Move back to the next vertex in the graph. If possible, form a new path starting at this vertex passing through vertices that were not already visited.

Step5: If this cannot be done, move back another vertex in the path and repeat the process described in step2.

Step6: Repeat this procedure, beginning at the last vertex visited, moving back up the path one vertex at a time, and form a new path until no more edges can be added.

This process ends, since the graph has a finite number of edges and is connected. Hence a spanning tree is produced.

Eg-1: Find the Depth First Traversal order for the given graph



Sol:

Step1: Choose the vertex v1 to be the root.

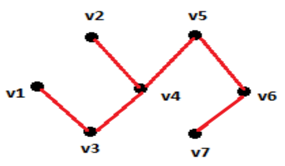
Step2: Form a path by successively adding edges incident with vertices which are not already in the path as long as possible. This procedure produces the path (v1, v3, v4, v5, v6, v7).

Step3: Now backtrack to v6. There is no path beginning at v6 containing vertices not already visited.

Step4: Similarly on backtracking to v5, we observe that there is no path.

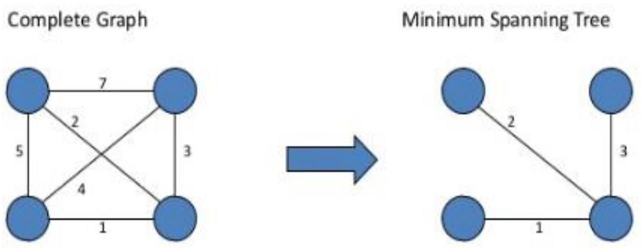
Step5: Backtrack to v4 and form the path (v4, v2).This produces the required spanning tree.

So the required path is v1, v3, v4, v5, v6, v7,v2 and the DFS spanning tree is shown below



Minimum Cost Spanning Tree (or) Minimal Spanning Tree:

Let 'G' be a weighted graph. A minimal spanning tree of G is defined as a spanning tree of 'G' with minimum weight.



Algorithms for finding minimal Spanning Tree:

1. Kruskal's
2. Prim's

1. **Kruskal's:**

   i/p: A connected weighted graph 'G'.
   o/p: A minimum cost(minimal) spanning tree 'T' for 'G'

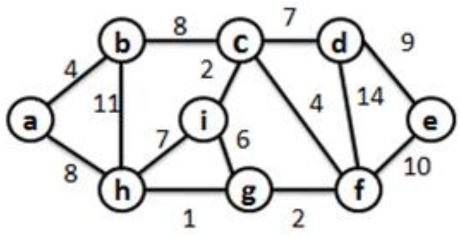
Procedure:

Step1: Choose an edge with minimal weight.

Step2: At each stage, choose (from the edges not yet chosen) the edge of lowest weight whose inclusion will not produce a cycle.

Step3: Stop the process of step2 when (n-1) edges are selected. These (n-1) edges contribute a minimum spanning tree for 'G'. Otherwise repeat step (2).

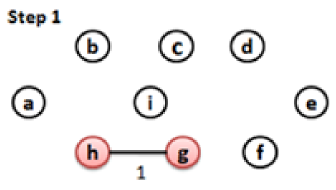
Eg: Find Minimum spanning tree for a graph in Fig using Kruskal's algorithm



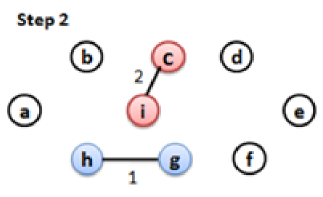
Sol: We first tabulate the edges with their weights(costs) in non-decreasing order.

Edge	(h,g)	(i,c)	(g,f)	(a,b)	(c,f)	(i,g)	(c,d)	(i,h)	(a,h)	(b,c)	(d,e)	(e,f)	(b,h)	(d,f)
Weight	1	2	2	4	4	6	7	7	8	8	9	10	11	14

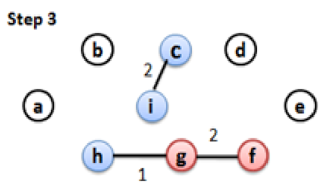
Step-1: Choose the edge (h,g) which has the minimal weight



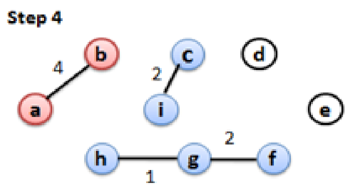
Step-2: There are two edges (i,c) and(g,f) with the next minimal cost. So choose both of them each in a step.So add the next edge (i,c).



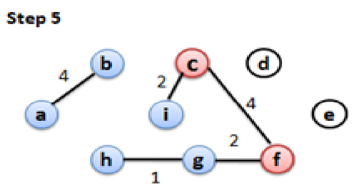
Step-3: Add the next edge (g,f).



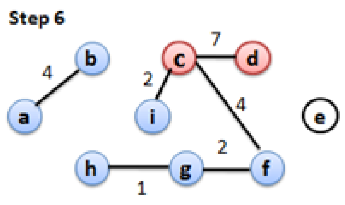
Step-4: Add the next edge (a,b).



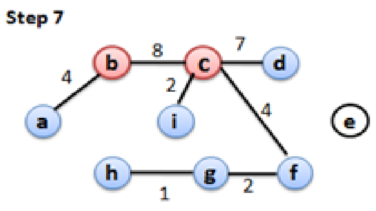
Step-5: Add the next edge (c,f).



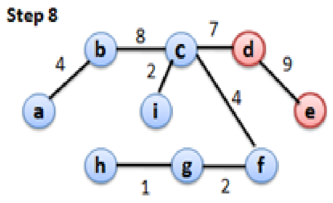
Step-6: The next edges with minimal cost are (i,g) and (i,h). But their inclusion will produce a cycle. So discard them and add the next edge (c,d).



Step-7: Add the next edge (b,c).



Step-8: The next edge with minimal cost is (a,h) but we discard it as its inclusion produces a cycle. So add the next edge (d,e).



Step-9: Discard all the remaining edges as their inclusion produces a cycle.Since the graph has 9 vertices it is enough to choose only 8 edges.Therefore we stop the procedure and the tree obtained in the Step-8 is the required Minimal Spanning tree and the Cost of the minimum spanning tree = 4+8+7+9+2+4+2+1 = 37

2. Prim's algorithm:

i/p: A connected weighted graph 'G'
o/p: A minimal spanning tree 'T'

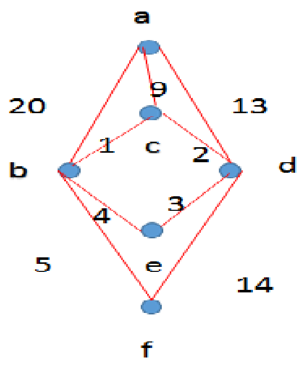
Procedure:

Step1: Select any vertex and choose the edge and minimum weight from 'G'.

Step2: At each stage choose the edge of smallest weight joining a vertex already included to vertex not yet included.

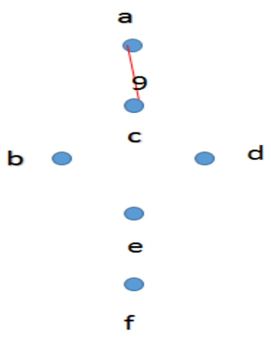
Step3: Continue until all vertices are included or If G has n vertices, then stop after (n-1) edges have been chosen otherwise repeat step(2).

Eg:

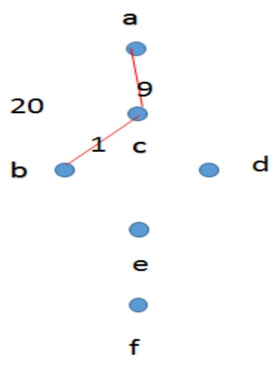


Sol:

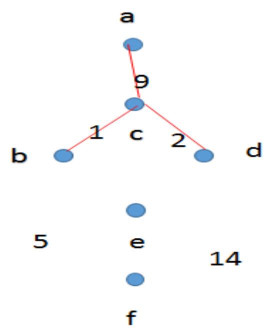
Step-1: Choose the vertex **'a'**. Now edges incident on 'a' are (a,b) , (a,c) and (a,d). Choose the edge with minimum weight i.e (a,c).



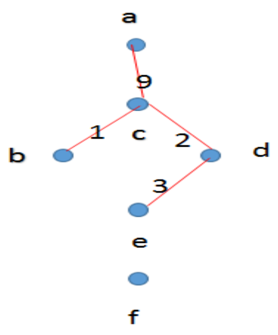
Step-2: The recent vertex is c. Now we have to choose the edge with minimum cost which are incident from both a and c which are already not included and which does not form a cycle. We have w(a,b)=20,w(a,d) =13,w(c,b) =1 and w(c,d) =2. So choose the edge (c,b)



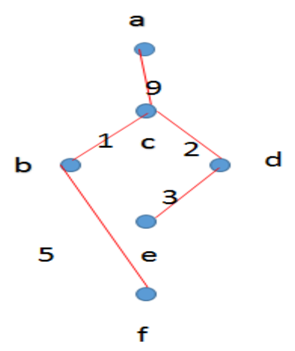
Step-3: The recent vertex is b. Now we have to choose the edge with minimum cost which are incident from a ,c and b which are already not included and which does not form a cycle. We have w(a,b)=20,w(a,d) =13,w(c,d) =2, w(be)=4 and w(b,f)=5. So choose the edge (c,d)



Step-4: The recent vertex is d. Now we have to choose the edge with minimum cost which are incident from a ,c,b and d which are already not included and which does not form a cycle. We have w(a,b)=20,w(a,d) =13, w(be)=4 ,w(b,f)=5,w(d,e)=3 and w(df)=14. So choose the edge (d,e)



Step-5: The recent vertex is e. Now we have to choose the edge with minimum cost which are incident from a ,c,b,d and e which are already not included and which does not form a cycle. We have w(a,b)=20,w(a,d) =13, w(be)=4 ,w(b,f)=5,w(df)=14. The edge with the next minimum cost is (b,e) but it is discarded as it forms a cycle.So choose the edge (b,f)



Step-6: The recent vertex is f. Now we have to choose the edge with minimum cost which are incident from a ,c,b,d,e and f which are already not included and which does not form a cycle. We have w(a,b)=20,w(a,d) =13, w(be)=4 and w(df)=14. We have to discard all these edges as they form a cycle .As all the vertices are covered we can stop this procedure and the tree obtained in the Step-5 is the required Minimal Spanning tree and the cost of the tree = 9+1+2+3+5 =20

.