MCA I SEMESTER
Mathematical Foundations of Computer Applications (MFCA): 20BM3101
Unit – 5: Graph Theory

Graph Theory: *Graph, Directed Graph, Multi Graph, Degree of vertex and their properties, Adjacency Matrix, Cycle Graph, Bipartite graphs, Isomorphism and Sub graphs, Trees and their properties, Spanning trees: DFS, BFS, Kruskal's Algorithm for finding minimal Spanning tree (Sections 5.1-5.4 of text book 2).*

Graph: A *graph* (or non directed graph) G is a pair of sets (V,E), where (i) V is a set of elements called *vertices* or *points* or *nodes* of G and (ii) E is a set of unordered pairs of vertices called *edges* of G. In this case we write $G=(V,E)$.

Directed graph: A *directed graph* (or *digraph*) G is a pair of sets (V,E), where (i) V is a set of elements called *vertices* or *points* or *nodes* of G and (ii) E is a set of ordered pairs of vertices called *directed edges* of G.

Note:

(i) If G is a graph, then $V(G)$ and $E(G)$ denote its sets of vertices and edges respectively.

(ii) In a graph G, an edge connected by the vertices u and v is denoted by $e=\{u,v\}$. In this case u and v are called *end points* of e. Also we say that e is *incident* on u and v.

(iii) In a digraph G, an edge connected from the vertex u to the vertex v is denoted by $e=(u,v)$. In this case u and v are called *initial* and *terminal points* of e respectively. Also we say that e is incident from u, incident to v.

Pictorial notation of a graph (or a digraph): Every graph $G=(V,E)$ can be represented by a pictorial diagram. In this diagram, every vertex is represented by a dot (or a small circle) and each edge is represented by curve which connects its end points.

Similarly, every digraph $G=(V,E)$ can be represented by a pictorial diagram. In this diagram, every vertex is represented by a dot (or a small circle) and each directed edge is represented by a directed curve which connects its initial and terminal points.

Note: In the diagram of a graph (or a digraph) the curve representing an edge should not pass through any points that represent vertices of the graph other than the end points.

Finite Graph: A graph $G=(V,E)$ is called a *finite graph* if it contains finite number of vertices and edges. That is, $|V|$ and $|E|$ are finite. In this case $|V|$ and $|E|$ are respectively called *order* and *size* of the graph.

Loop: In a graph (or digraph) G, an edge between a vertex and itself is called a *loop* (*self-loop*).

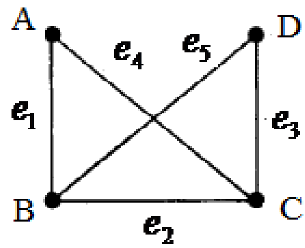
Multiple edges: Two or more edges of a graph are called *multiple edges* if they have same end points.

Multi graph: A graph G is called a *multi graph* if it may contain loops and multiple edges.

Simple graph: A graph without loops and multiple edges is called a *simple graph*.

Adjacent vertices or neighbors: Let $G=(V,E)$ be a graph (or directed graph). Then two vertices u,v are said to be *adjacent* or *neighbors* if there is an edge (or a directed edge) between u and v.

Example 1: Consider the graph $G=(V,E)$, where $V=\{A,B,C,D\}$ and $E=\{\{A,B\},\{A,C\},\{B,C\},\{B,D\},\{C,D\}\}$



Here

(i) A,B,C,D are vertices and $\{A,B\},\{A,C\},\{B,C\},\{B,D\},\{C,D\}$ are non directed edges.

(ii) $|V|=4$ and $|E|=5$

(iii) G is a finite graph with order 4 and size 5

(iv) A,B are the end points of the edge e_1 and A,B are adjacent vertices (or neighbors)

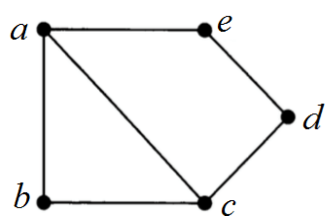
(v) The edge e_1 incident on A and B

(vi) A,C are the end points of the edge e_4 and A,C are adjacent vertices (or neighbors)

(vii) The edge e_4 incident on A and C

(viii) There are no loops and multiple edges and hence it is a simple graph

Example 2: Consider the graph $G=(V,E)$, where $V=\{a,b,c,d,e\}$ and $E=\{\{a,b\},\{a,c\},\{a,e\},\{b,c\},\{c,d\},\{d,e\}\}$



Here

(i) a,b,c,d,e are vertices and $\{a,b\}$, $\{a,c\}$, $\{a,e\}$, $\{b,c\}$, $\{c,d\}$, $\{d,e\}$ are non directed edges.

(ii) $|V|=5$ and $|E|=6$

(iii) G is a finite graph with order 5 and size 6

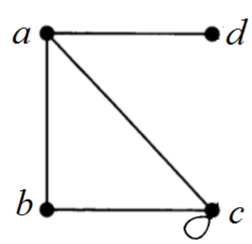
(iv) c,d are the end points of the edge $\{c,d\}$ and c,d are adjacent vertices (or neighbors)

(v) The edge $\{c,d\}$ incident on c and d

(vi) a and d are not adjacent vertices (or not neighbors)

(vii) There are no loops and multiple edges and hence it is a simple graph

Example 3: Consider the graph $G=(V,E)$, where $V=\{a,b,c,d\}$ and $E=\{\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{c,c\}\}$



Here

(i) a,b,c,d are vertices and $\{a,b\}$, $\{a,c\}$, $\{a,d\}$, $\{b,c\}$, $\{c,c\}$ are non directed edges.

(ii) $|V|=4$ and $|E|=5$

(iii) G is a finite graph with order 4 and size 5

(iv) a,b are the end points of the edge $\{a,b\}$ and a,b are adjacent vertices (or neighbors)

(v) The edge $\{a,b\}$ incident on a and b

(vi) c and d are not adjacent vertices (or not neighbors)

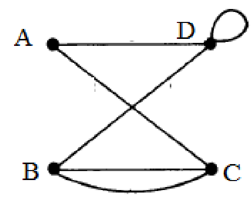
(vii) b and d are not adjacent vertices (or not neighbors)

(viii) There is a loop at the vertex c

(ix) There are no multiple edges

(x) It is not a simple graph

Example 4: Consider the graph $G=(V,E)$, where $V=\{A,B,C,D\}$ and
$E=\{\{A,C\},\{A,D\},\{B,C\},\{B,C\},\{B,D\},\{D,D\}\}$



Here

(i) A,B,C,D are vertices and $\{A,C\},\{A,D\},\{B,C\},\{B,C\},\{B,D\},\{D,D\}$ are non directed edges.

(ii) $|V|=4$ and $|E|=6$

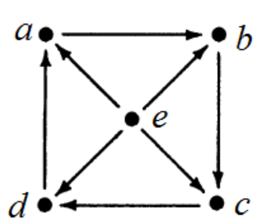
(iii) G is a finite graph with order 4 and size 6

(iv) There is a loop at the vertex D

(v) There are multiple edges between the vertices B and C

(vi) It is not a simple graph

Example 5: Consider the graph $G=(V,E)$, where $V=\{a,b,c,d,e\}$ and
$E=\{(a,b),(b,c),(c,d),(d,a),(e,a),(e,b),(e,c),(e,d)\}$



Here

(i) a,b,c,d,e are vertices and (a,b), (b,c), (c,d), (d,a), (e,a), (e,b), (e,c), (e,d) are directed edges.

(ii) $|V|=5$ and $|E|=8$

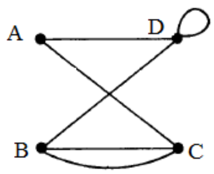
(iii) G is a finite graph with order 5 and size 8

(iv) a,b are the initial and terminal points of the directed edge (a,b) and a, b are adjacent vertices (or neighbors)

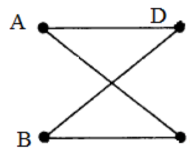
(v) The edge (a,b) incident from a and incident to b

(vi) There are no loops and multiple edges and hence it is a simple digraph

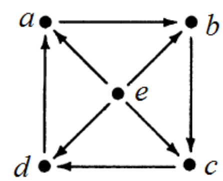
Underlying simple graph: If $G=(V,E)$ is a graph, then the graph obtained from G by ignoring all the loops and considering only one edge in case of multiple edges is called the *underlying simple graph* of G.
Underlying non directed graph: If $G=(V,E)$ is a digraph, then the non directed graph obtained from G by ignoring the direction of the edges is called the *underlying non directed graph* of G.



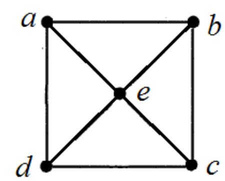
Graph G



Underlying simple graph of G



Digraph G



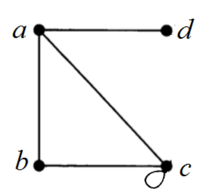
Underlying non directed graph of G

Degree of a vertex: The *degree* of a vertex v in a graph G is defined as the number of edges in G which contain v as an end point. It is denoted by $\deg(v)$. In the counting of the degree of a vertex v, a loop at v is considered twice.
In degree: The *in degree* of a vertex v in a digraph G is defined as the number of edges in G which contain v as a terminal point. It is denoted by $\deg^{+}(v)$.
Out degree: The *out degree* of a vertex v in a digraph G is defined as the number of edges in G which contain v as an initial point. It is denoted by $\deg^{-}(v)$.
Isolated vertex: A vertex v of a graph G, is called an *isolated vertex* if $\deg(v)=0$

Min degree and max degree: The minimum of all the degrees of the vertices of a graph G is called *min degree* of G and is denoted by $\delta(G)$. The maximum of all the degrees of the vertices of a graph G is called *max degree* of G and is denoted by $\Delta(G)$.

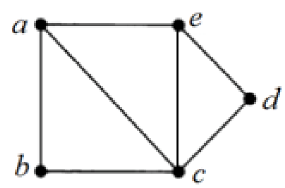
Degree sequence: In a graph G, the sequence of the degrees of the vertices in the increasing order is called the degree sequence of G. If $v_1, v_2, v_3, \cdots v_n$ are the vertices of G in the increasing order of their degrees and $d_1, d_2, d_3, \cdots d_n$ are respectively the degrees of the vertices; that is $\deg(v_i)=d_i$ for $i=1,2,3, \cdots n$ and $d_1 \le d_2 \le d_3 \le \cdots \le d_n$, then the degree sequence of G is denoted by $(d_1, d_2, d_3, \cdots d_n)$

Example 6: Consider the graph $G=(V,E)$, where $V=\{a,b,c,d\}$ and
$E=\{\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{c,c\}\}$



Here
(i) $\deg(a)=3,\ \deg(b)=2,\ \deg(c)=4,\ \deg(d)=1$
(ii) The degree sequence is $(1,\ 2,\ 3,\ 4)$
(iii) There is no isolated vertex
(iv) Minimum degree, $\delta(G)=1$
(v) Maximum degree, $\Delta(G)=4$
(vi) Sum of the degrees is equal to the twice the number of edges; that is, $\sum_{i=1}^{4}\deg(v_i)=2|E|=10$

Example 7: Consider the graph $G=(V,E)$, where $V=\{a,b,c,d,e\}$ and
$E=\{\{a,b\},\{a,c\},\{a,e\},\{b,c\},\{c,d\}\{c,e\},\{d,e\}\}$



Here

(i) $\deg(a)=3,\ \deg(b)=2,\ \deg(c)=4,\ \deg(d)=2,\ \deg(e)=3$

(ii) The degree sequence is $(2,\ 2,\ 3,\ 3,\ 4)$

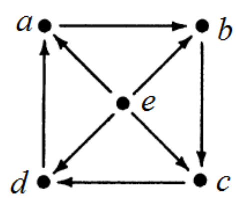
(iii) There is no isolated vertex

(iv) Minimum degree, $\delta(G)=2$

(v) Maximum degree, $\Delta(G)=4$

(vi) Sum of the degrees is equal to the twice the number of edges; that is, $\sum_{i=1}^{4}\deg(v_i)=2|E|=14$

Example 8: Consider the digraph $G=(V,E)$, where $V=\{a,b,c,d,e\}$ and
$E=\{(a,b),(b,c),(c,d),(d,a),(e,a),(e,b),(e,c),(e,d)\}$



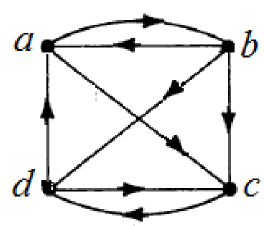
Here

(i) In degrees: $\deg^+(a)=2,\ \deg^+(b)=2,\ \deg^+(c)=2,\ \deg^+(d)=2,\ \deg^+(e)=0$

(ii) Out degrees: $\deg^-(a)=1,\ \deg^-(b)=1,\ \deg^-(c)=1,\ \deg^-(d)=1,\ \deg^-(e)=4$

(iii) Sum of the in degrees, sum of the out degrees and the number of the directed edges are same; that is, $\sum_{i=1}^{5}\deg^+(v_i)=\sum_{i=1}^{5}\deg^-(v_i)=|E|=8$

Example 9: Consider the digraph $G=(V,E)$, where $V=\{a,b,c,d\}$ and
$E=\{(a,b),(a,c),(b,a),(b,c),(b,d),(c,d),(d,a),(d,c)\}$



Here

(i) In degrees: $\deg^+(a)=2,\ \deg^+(b)=1,\ \deg^+(c)=3,\ \deg^+(d)=2$

(ii) Out degrees: $\deg^-(a)=2,\ \deg^-(b)=3,\ \deg^-(c)=1,\ \deg^-(d)=2$

(iii) Sum of the in degrees, sum of the out degrees and the number of the directed edges are same; that is, $\sum_{i=1}^{5}\deg^+(v_i)=\sum_{i=1}^{5}\deg^-(v_i)=|E|=8$

Regular & Cubic graphs: A graph G is called a *regular graph* of degree k (or $k-regular$) if the degree of every vertex is k, where k is non-negative integer. In particular a regular graph of degree 3 is called a *cubic graph.*

Complete graph: A simple graph G is called a *complete graph* if there is an edge between every pair of vertices of G. A complete graph with n vertices is denoted by K_n.

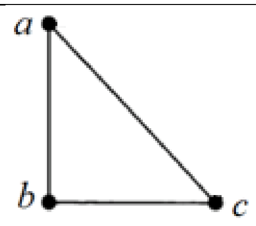
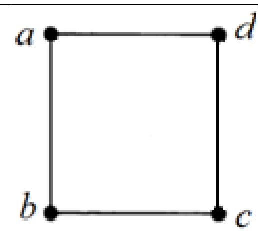
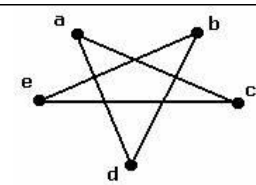
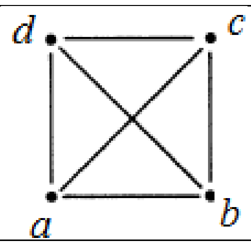
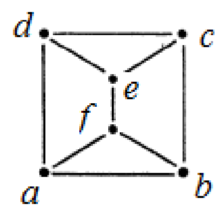
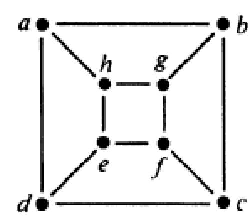
Note:

(i) In a complete graph K_n, the degree of each vertex is $n-1$

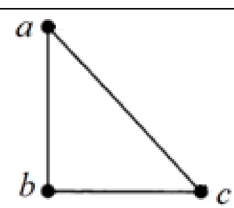
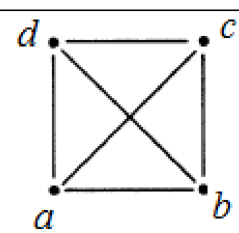
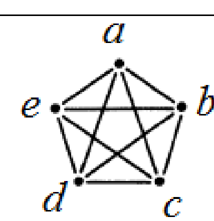
(ii) Every complete graph K_n is a regular graph of degree $n-1$

(iii) The number of edges in a complete graph K_n is ${}^nC_2$ or $\dfrac{n(n-1)}{2}$

Example 10: Examples for regular and cubic graphs

		
Degree of each vertex is 2	Degree of each vertex is 2	Degree of each vertex is 2
Regular graph of degree 2	Regular graph of degree 2	Regular graph of degree 2
Or 2-regular	Or 2-regular	Or 2-regular
		
Degree of each vertex is 3	Degree of each vertex is 3	Degree of each vertex is 3
Regular graph of degree 3	Regular graph of degree 3	Regular graph of degree 3
It is cubic graph	It is cubic graph	It is cubic graph

Example 11: Examples for complete graph K_n

				
Complete graph K_1	Complete graph K_2	Complete graph K_3	Complete graph K_4	Complete graph K_5

Theorem 1: (First theorem of graph theory or Sum of degrees theorem)

If $V=\{v_1,v_2,v_3,\cdots v_n\}$ is the vertex set of a non directed graph G, then $\sum_{i=1}^{n}\deg(v_i)=2|E|$

If G is a directed graph, then $\sum_{i=1}^{n}\deg^{+}(v_i)=\sum_{i=1}^{n}\deg^{-}(v_i)=|E|$

Proof: **Non-directed graph case:**
Let $V=\{v_1,v_2,v_3,\cdots v_n\}$ be the vertex set of a graph $G=(V,E)$ and $e=\{a,b\}$ be an edge.

Then the edge e contributes 1 time each in the computation of the degrees of a and b.
Therefore, every edge contributes 2 times in the computation of the sum of the degrees of all the vertices of G. Hence the sum of the degrees of all the vertices is same as the twice the number of edges. That is $\sum_{i=1}^{n}\deg(v_i)=2|E|$.

Directed graph case:
Let $V=\{v_1,v_2,v_3,\cdots v_n\}$ be the vertex set of a digraph $G=(V,E)$ and $e=(a,b)$ be a directed edge.

Then the edge e contributes 1 time each in the computation of the out degree of a and in degree of b.
Therefore, every edge contributes 1 time each in the computation of the sum of the in degrees and the sum of the out degrees of all the vertices of G. Hence the sum of the in degrees and sum of the out degrees of all the vertices are same as the number of edges. That is $\sum_{i=1}^{n}\deg^{+}(v_i)=\sum_{i=1}^{n}\deg^{-}(v_i)=|E|$.

Theorem 2: In any non directed graph there is even number of vertices of odd degree
Proof: Let $G=(V,E)$ be a graph. Let V_E and V_O be sets of vertices of even and odd degrees respectively so that $V_E\cup V_O=V$ and $V_E\cap V_O=\phi$.

By the first theorem of graph theory, we have $\sum_{v\in V}\deg(v)=2|E|$

But we can write, $\sum_{v\in V}\deg(v)=\sum_{v\in V_E}\deg(v)+\sum_{v\in V_O}\deg(v)$

Therefore, $\sum_{v\in V_O}\deg(v)=\sum_{v\in V}\deg(v)-\sum_{v\in V_E}\deg(v)$

$$=2|E|-\sum_{v\in V_E}\deg(v)$$

Since $2|E|$ and $\sum_{v\in V_E}\deg(v)$ are even numbers, we have $\sum_{v\in V_O}\deg(v)$ is also even number

If the number of elements in V_O is odd then the sum of odd number of odd integers is odd, which leads a contradiction. Therefore, the number of elements in V_O is even; that is, there are even number of vertices of odd degree.

Theorem 3: If G is a non directed graph, then
$\delta(G)\,|V| \le \sum_{v\in V(G)} \deg(v) \le \Delta(G)|V|$ or $\delta(G)\,|V| \le 2|E| \le \Delta(G)|V|$.
In particular if G is $k-$regular graph, then $k|V| = \sum_{v\in V(G)} \deg(v) = 2|E|$.

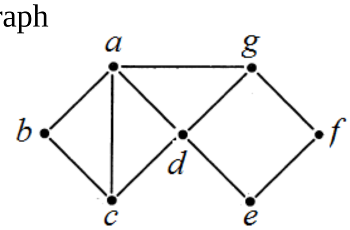
Proof: We know that $\sum_{v\in V(G)} \deg(v) = 2|E|$
Also, $\delta(G) = \text{Min}\{\deg(v) \mid v \in V\}$ and $\Delta(G) = \text{Max}\{\deg(v) \mid v \in V\}$
So that we have $\delta(G) \le \deg(v) \le \Delta(G)$ for all $v \in V$
Therefore, $\sum_{v\in V} \delta(G) \le \sum_{v\in V} \deg(v) \le \sum_{v\in V} \Delta(G)$
That is, $\delta(G)\,|V| \le \sum_{v\in V(G)} \deg(v) \le \Delta(G)|V|$ or $\delta(G)\,|V| \le 2|E| \le \Delta(G)|V|$
If G is $k-$regular graph, then $\deg(v) = \delta(G) = \Delta(G) = k$ for all $v \in V$
Therefore, $k|V| \le 2|E| \le k|V|$ or $2|E| = k|V|$
Hence $k|V| = 2|E| = \sum_{v\in V(G)} \deg(v)$

Path: In a graph G, a sequence of edges of the form $\{v_0, v_1\}, \{v_1, v_2\}, \{v_2, v_3\}, \cdots \{v_{n-1}, v_n\}$ is called a *path* from v_0 to v_n. It is denoted by $v_0 - v_1 - v_2 - v_3 - \cdots v_{n-1} - v_n$ and simply designated as $v_0 - v_n$ path. Here the *end points* of the path v_0 and v_n are respectively called *initial* and *terminal points* of the path.

Note:
(i) In a path of a graph, vertices and edges may be repeated.
(ii) Every path in a graph is itself a graph.

Closed and Open paths: A path in a graph G is called a *closed path* if its end points are equal. Otherwise the path is called *open path*
Length of a path: The number of edges in a path is called the *length* of the path
Simple path: A path in which all the edges and vertices are distinct except possibly the end points is called a *simple path*.
Note:
(i) An open simple path of length n has $n+1$ distinct vertices and n distinct edges.
(ii) A closed simple path of length n has n distinct vertices and n distinct edges.
(iii) The *trivial path* is a simple closed path of length zero.

Example 12: Consider the following graph



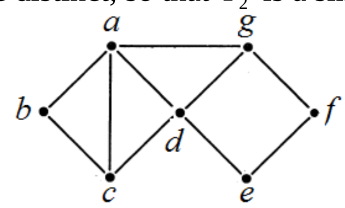
Here
(i) $P_1 : b - c - a - d - g$ is a path from b to g (or $b - g$ path)
(ii) Number of edges in P_1 is 4, so that the length of P_1 is 4
(iii) The initial and terminal points of P_1 are b and g
(iv) The end points of P_1 are different, so that P_1 is an open path
(v) All the edges and vertices of P_1 are distinct, so that P_1 is a simple path

(vi) $P_2 : b-a-g$ is another path from b to g (or $b-g$ path)

(vii) Number of edges in P_2 is 2, so that the length of P_2 is 2

(viii) The end points of P_2 are different, so that P_2 is an open path

(ix) All the edges and vertices of P_2 are distinct, so that P_2 is a simple path



(x) $P_3 : b-a-d-c-a-g$ is another $b-g$ path of length 5

(xi) The end points of P_3 are different, so that P_3 is an open path

(xii) The vertices of P_3 are not distinct, so that P_3 is not a simple path

(xiii) $P_4 : b-a-d-c-a-d-g$ is another $b-g$ path of length 6

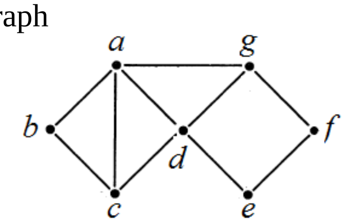
(xiv) The end points of P_4 are different, so that P_4 is an open path

(xv) The vertices and edges of P_4 are not distinct, so that P_4 is not a simple path

Circuit: A closed path of length greater than 1 with no repeated edges is called a *circuit*.
Note: A circuit may have repeated vertices other than the end points.
Cycle: A closed path of length greater than 1 with no repeated edges and with no other repeated vertices except the end points is called a *cycle*.
Note:

(i) A circuit may have repeated vertices other than the end points but no repeated edges
(ii) A cycle have no repeated vertices other than the end points and no repeated edges
(iii) Every cycle is a circuit but a circuit may not be a cycle.
(iv) A loop is a cycle of length 1
(v) In a graph, a cycle that is not a loop must have length at least 3, but there may be cycles of length 2 in a multi graph.

Example 13: Consider the following graph



Here

(i) $P_1 : b-a-d-c-b$ is a closed path of length 4

(ii) All the edges and vertices (except the end points) of P_1 are distinct, so that P_1 is a cycle and hence it is a circuit

(iii) $P_2 : b-a-d-g-a-c-b$ is a closed path of length 6

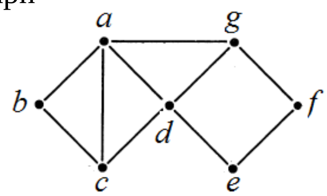
(iv) All the edges of P_2 are distinct, so that P_2 is a circuit

(v) The vertices of P_2 are not distinct (a is repeated), so that P_2 is not a cycle

(vi) $P_3 : b-a-d-g-a-d-c-b$ is a closed path of length 7

(vii) The edges of P_3 are not distinct, so that P_3 is not a circuit and hence it is not a cycle

Edge-disjoint paths: Two paths in a graph are said to be *edge-disjoint* if they do not share any common edges.
Vertex-disjoint paths: Two paths in a graph are said to be *vertex-disjoint* if they do not share any common vertices.
Example 14: Consider the following graph



Here

(i) $P_1: a-d-e-f$ is a $a-f$ path of length 3
(ii) $P_2: b-c-d-g$ is a $b-g$ path of length 3
(iii) P_1 and P_2 have no common edges, so that P_1 and P_2 are edge disjoint paths
(iv) P_1 and P_2 have a common vertex d, so that P_1 and P_2 are not vertex disjoint paths

(v) $P_3: b-c-d-e$ is a $b-e$ path of length 3
(vi) $P_4: a-g-f$ is a $a-f$ path of length 2
(vii) P_3 and P_4 have no common edges, so that P_1 and P_2 are edge disjoint paths
(viii) P_3 and P_4 have no common vertices, so that P_1 and P_2 are vertex disjoint paths

(ix) $P_5: a-d-e-f$ is a $a-f$ path of length 3
(x) P_3 and P_5 have a common edge and common vertices, so that P_1 and P_2 are neither edge disjoint nor vertex disjoint paths

Theorem 4: In a graph G, every $u-v$ path contains a simple $u-v$ path
Proof: Let P be a $u-v$ path of length n.
If P is closed path, then P contains a trivial path of length 0

If P is open path, then we prove this theorem by applying induction on n
If $n=0$ then P is a trivial path and hence it is simple
If $n=1$ then P has only one edge and hence it is simple

Assume that every $u-v$ path of length less than n contains a simple $u-v$ path
Now we prove the theorem for a $u-v$ path of length n

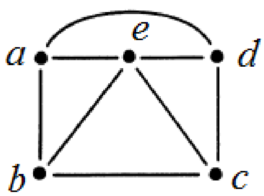
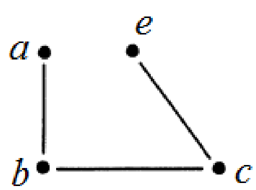
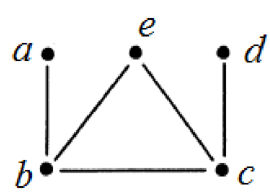
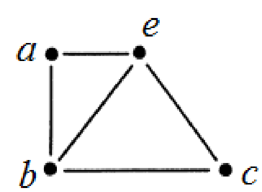
Let $P: u=v_0-v_1-v_2 \cdots -v_i-v_{i+1} \cdots -v_j-v_{j+1} \quad \cdots \quad v_{n-1}-v_n=v$ be a $u-v$ path of length n

If P is a simple path, then the theorem is proved
If P is not a simple path, then it has at least one repeated vertex
Suppose that $v_i=v_j$ for some $i<j$, then P has a closed path from v_i to v_j
If this closed path removed from P, then an open path P' is obtained with length less than n
Therefore by induction hypothesis, P' contains a simple $u-v$ path and hence P contains the same
Thus by the mathematical induction, every $u-v$ path of any length n contains a simple $u-v$ path

Edge labeling: If G is a graph then any mapping from $E(G) \to D$, where D is some domain of labels, is called an *edge labeling*.
Vertex labeling: If G is a graph then any mapping from $V(G) \to D$, where D is some domain of labels, is called a *vertex labeling*.

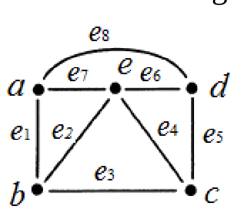
Subgraph, Spanning subgraph and Induced subgraph: Let G be a graph.
(i) A graph H is called a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$
(ii) A graph H is called a *spanning subgraph* of G if $V(H) = V(G)$ and $E(H) \subseteq E(G)$
(iii) A graph H is called a subgraph of G *induced by a set* W, where $W \subseteq V(G)$ if $V(H) = W$ and $E(H)$ is the set of all edges of G with end points belong to W

Example 15:

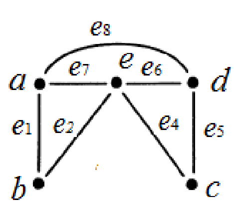
			
Graph G	Subgraph H_1	Spanning subgraph H_2	Induced subgraph H_3
	$V(H_1) \subseteq V(G)$ and $E(H_1) \subseteq E(G)$	$V(H_2) = V(G)$ and $E(H_2) \subseteq E(G)$	Induced by $W = \{a,b,c,e\} \subseteq V(G)$

Removal of edges and vertices: Let G be a graph.
(i) If $e \in E(G)$ then $G - e$ is the graph obtained from G by deleting the edge e
(ii) In general if $e_1, e_2, \cdots e_k \in E(G)$ then $G - \{e_1, e_2, \cdots e_k\}$ is the graph obtained from G by deleting the edges $e_1, e_2, \cdots e_k$
(iii) If $v \in V(G)$ then $G - v$ is the graph obtained from G by removing the vertex v and all the edges connected to v
(iv) In general if $v_1, v_2, \cdots v_k \in V(G)$ then $G - \{v_1, v_2, \cdots v_k\}$ is the graph obtained from G by removing the vertices $v_1, v_2, \cdots v_k$ and all the edges connected to these vertices

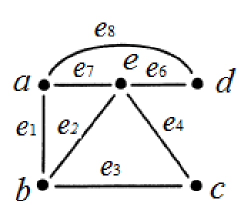
Example 16: Consider the graph G



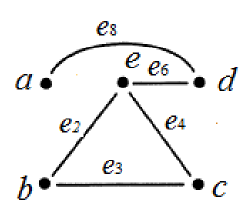
Graph G



G - e_3

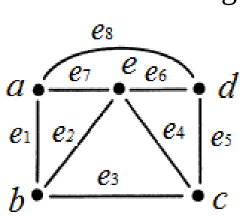


G - e_5

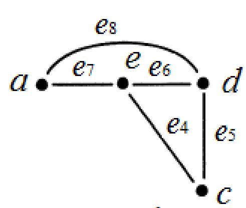


G - $\{e_1, e_5, e_7\}$

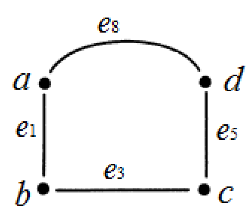
Example 17: Consider the graph G



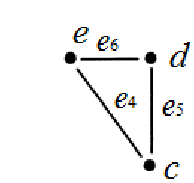
Graph G



G - b



G - e

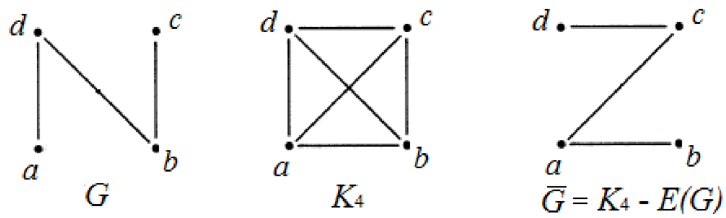


G - $\{a, b\}$

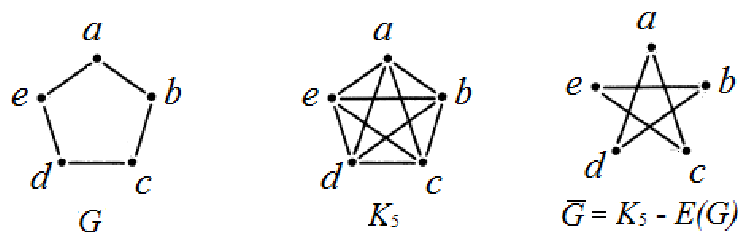
Complement of a graph: If G is a simple graph with n vertices, then the *complement* of G is denoted by $\overline{G}$ and is given by $\overline{G} = K_n - E(G)$ where K_n is the complete graph with n vertices.

Complement of a subgraph: If H is a subgraph of a graph G, then the *complement* of H is denoted by $\overline{H}$ and is given by $\overline{H} = G - E(H)$.

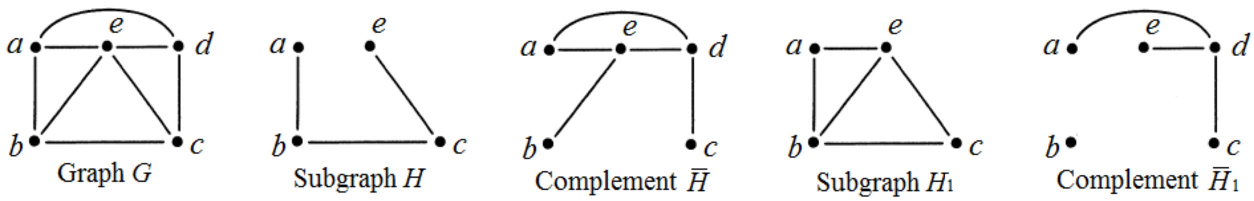
Example 18: Consider the following graph G and its complement $\overline{G}$



Example 19: Consider the following graph G and its complement $\overline{G}$



Example 20: Consider the subgraphs H and H_1 of the graph G and the complements $\overline{H}$ and $\overline{H}_1$



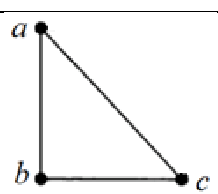
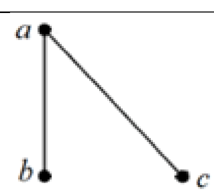
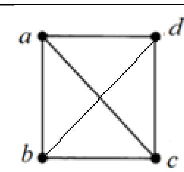
Cycle graph: A *cycle graph* of order n is a graph whose edges form a cycle of length n. It is denoted by C_n.

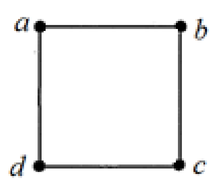
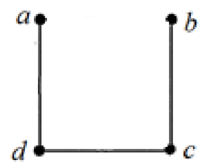
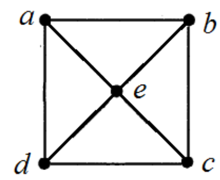
Wheel graph: A *wheel graph* of order n is a graph obtained by joining a single new vertex to each vertex of a cycle graph of order $n-1$. It is denoted by W_n.

Path graph: A *path graph* of order n is a graph obtained by removing an edge from a cycle graph of order n. It is denoted by P_n.

Null graph: A *null graph* of order n is a graph with n vertices and no edges. It is denoted by N_n.

Example 21:

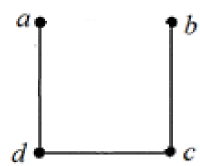
		
Cycle graph C_3	Path graph P_3	Wheel graph W_4

		
Cycle graph C_4	Path graph P_4	Wheel graph W_5

Bipartite graph: A graph G is called a *bipartite graph* if its vertex set $V(G)$ is partitioned in to two sets M and N in such a way that each edge joins a vertex in M to a vertex in N

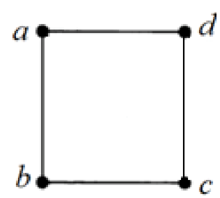
Complete bipartite graph: A graph G is called a *complete bipartite* graph if its vertex set $V(G)$ is partitioned in to two sets M and N in such a way that there is an edge between each vertex in M to each vertex in N. If $|M| = m$, $|N| = n$ and $m \leq n$ then the complete bipartite graph is denoted by $K_{m,n}$. In particular the complete bipartite graph $K_{1,n}$ is called a *star graph.*

Example 22: Consider the following graph



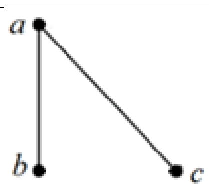
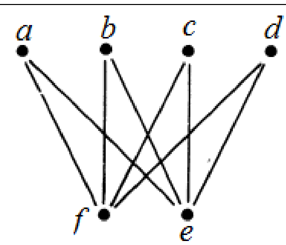
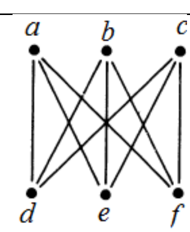
It is a bipartite graph with $M = \{a, c\}$ and $N = \{b, d\}$. Here each edge of G is connected between a vertex of M and a vertex of N

Example 23: Consider the following graph



It is a complete bipartite graph with $M = \{a, c\}$ and $N = \{b, d\}$. Here there is an edge in G between each vertex of M to each vertex of N. It is $K_{2,2}$

Example 24: The following are complete bipartite graphs

		
Complete bipartite graph with $M = \{a\}$ and $N = \{b, c\}$	Complete bipartite graph with $M = \{e, f\}$ and $N = \{a, b, c, d\}$	Complete bipartite graph with $M = \{a, b, c\}$ and $N = \{d, e, f\}$
$K_{1,2}$	$K_{2,4}$	$K_{3,3}$

Adjacency matrix: Let $v_1, v_2, \cdots v_n$ be the vertices of a graph G. Then the *adjacency matrix* for this ordering of the vertices of G is denoted by A_G and is defined as $A_G = (a_{ij})_{n\times n}$, where

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

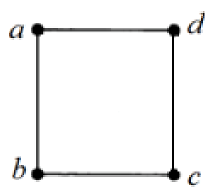
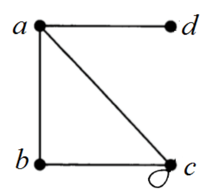
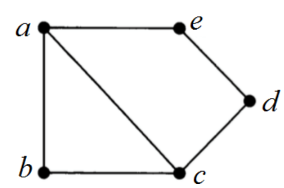
Note:

(i) Adjacency matrix of a graph is always symmetric

(ii) $a_{ii} = 1$ if and only if there is a loop at the vertex v_i

(iii) If we change the ordering of the vertices, then the entries of the adjacency matrix will be rearranged

Example 25: Consider the graph G

Graph G			
Adjacency matrix of G	$A_G = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$	$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$	$A_G = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$

Isomorphism of graphs: Two graphs G and G' are said to be *isomorphic* if there exists a function $f: V(G) \to V(G')$ such that

(i) f is one-one

(ii) f is onto

(iii) $\{u,v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(G')$ for all $u,v \in V(G)$; that is,
There is an edge between u and v if and only if there is an edge between $f(u)$ and $f(v)$. In other words we say that preserves adjacency.

Note:

(i) A function f satisfying the above three conditions is called an *isomorphism* from G to G'

(ii) If G and G' are isomorphic graphs then there may be more than one isomorphisms from G to G'

(iii) If G and G' contains n elements each, then there are $n!$ bijection (one-one and onto) functions from G to G'. Not every bijection function satisfies the above condition (iii). If at least one bijection satisfies the condition (iii) then G and G' are isomorphic.

Deductions from isomorphic graphs: If G and G' are isomorphic graphs with an isomorphism f from G to G', then we have following necessary properties.

(i) $|V(G)| = |V(G')|$ that is, number of vertices of G and G' are equal

(ii) $|E(G)| = |E(G')|$ that is, number of edges of G and G' are equal

(iii) The degree sequences of G and G' are equal

(iv) The number of loops in G and G' are equal

(v) The number $k-$cycles in G and G' are equal

(vi) The subgraphs induced by the vertices of degree k in each graph are isomorphic

(vii) $\deg(v) = \deg[f(v)]$ for all $v \in V(G)$

(viii) There is a loop at v if and only if there is a loop at $f(v)$ for all $v \in V(G)$

(ix) $v_0 - v_1 - v_2 - \cdots v_{k-1} - v_k$ is a cycle of length k in G if and only if $f(v_0) - f(v_1) - f(v_2) - \cdots f(v_{k-1}) - f(v_k)$ is a cycle of length k in G'

(x) The adjacency matrix A_G of G in the vertex ordering $v_1, v_2, \cdots v_n$ is equal to the adjacency matrix A_{G^1} of G' in the vertex ordering $f(v_1), f(v_2), \cdots f(v_n)$ which is equivalent to the edge preserving condition

Note:

(i) If any one of the properties from (i) to (vi) fails, then the graphs are not isomorphic.

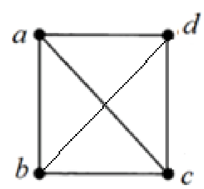
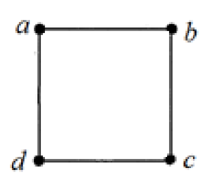
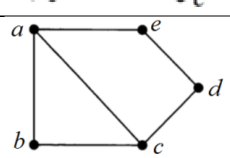
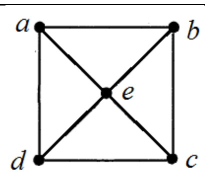
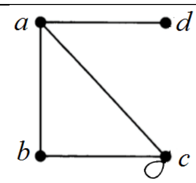
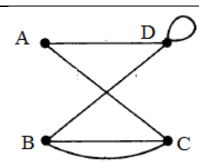
(ii) If any one of the properties from (vii) to (x) fails, then the function f is not an isomorphism. In this case we need to find another bijection for an isomorphism.

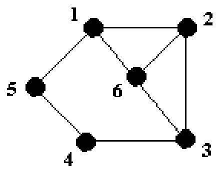
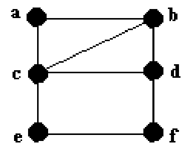
Alternate definitions for isomorphism of graphs:

1. Two graphs G and G' are said to be *isomorphic* if there exists a function $f : V(G) \to V(G')$ such that
   (i) f is one-one and onto
   (ii) The adjacency matrix A_G of G in the vertex ordering $v_1, v_2, \cdots v_n$ is equal to the adjacency matrix A_{G^1} of G' in the vertex ordering $f(v_1), f(v_2), \cdots f(v_n)$

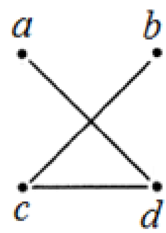
2. Two graphs G and G' are said to be *isomorphic* if and only if the complement graphs of G and G' are isomorphic

Example 26: The following graphs are not isomorphic

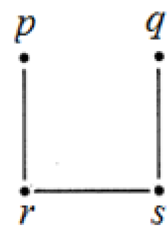
	Graph G	Graph G'	
1			(i) $\lvert V(G)\rvert = 4, \lvert V(G')\rvert = 4$ (ii) $\lvert E(G)\rvert = 6, \lvert E(G')\rvert = 4$ not equal Not isomorphic
2			(i) $\lvert V(G)\rvert = 5, \lvert V(G')\rvert = 5$ (ii) $\lvert E(G)\rvert = 6, \lvert E(G')\rvert = 6$ (iii) Degree sequences: (2,2,2,3,3) and (3,3,3,3,4) not equal Not isomorphic
3			(i) $\lvert V(G)\rvert = 4, \lvert V(G')\rvert = 4$ (ii) $\lvert E(G)\rvert = 5, \lvert E(G')\rvert = 5$ (iii) There is a vertex in G of degree 1, but there is no vertex in G' of degree 1 Not isomorphic

4	 	(i) $\lvert V(G)\rvert = 6, \lvert V(G')\rvert = 6$ (ii) $\lvert E(G)\rvert = 8, \lvert E(G')\rvert = 8$ (iii) There is no vertex in G of degree 4, but there is a vertex in G' of degree 4 Not isomorphic

Example 27: Show that the following graphs are isomorphic



G



G'

Solution: Here

(i) $\lvert V(G)\rvert = 4$ and $\lvert V(G')\rvert = 4$ $\quad \therefore \lvert V(G)\rvert = \lvert V(G')\rvert$

(ii) $\lvert E(G)\rvert = 3$ and $\lvert E(G')\rvert = 3$ $\quad \therefore \lvert E(G)\rvert = \lvert E(G')\rvert$

(iii) In G, $\deg(a) = 1,\ \deg(b) = 1,\ \deg(c) = 2,\ \deg(d) = 2$ and the degree sequence is $(1,\ 1,\ 2,\ 2)$
In G', $\deg(p) = 1,\ \deg(q) = 1,\ \deg(r) = 2,\ \deg(s) = 2$ and the degree sequence is $(1,\ 1,\ 2,\ 2)$
$\therefore$ The degree sequences are equal

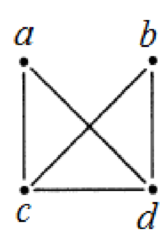
(iv) Define $f : V(G) \to V(G')$ such that $f(a) = p,\ f(b) = q,\ f(c) = s,\ f(d) = r$
Then f is one-one and onto

(v) In G and G', for the vertex ordering a, b, c, d and $f(a), f(b), f(c), f(d)$ respectively, the adjacency matrices are given by

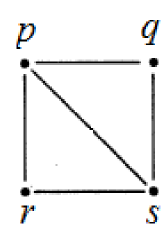
$$A_G = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad A_{G^1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Therefore, $A_G = A_{G^1}$. Hence G and G' are isomorphic

Example 28: Show that the following graphs are isomorphic



G



G'

Solution: Here

(i) $\lvert V(G)\rvert = 4$ and $\lvert V(G')\rvert = 4$ $\quad \therefore \lvert V(G)\rvert = \lvert V(G')\rvert$

(ii) $\lvert E(G)\rvert = 5$ and $\lvert E(G')\rvert = 5$ $\quad \therefore \lvert E(G)\rvert = \lvert E(G')\rvert$

(iii) In G, $\deg(a) = 2,\ \deg(b) = 2,\ \deg(c) = 3,\ \deg(d) = 3$ and the degree sequence is $(2,\ 2,\ 3,\ 3)$

In G', $\deg(p)=3$, $\deg(q)=2$, $\deg(r)=2$, $\deg(s)=3$ and the degree sequence is $(2, 2, 3, 3)$

$\therefore$ The degree sequences are equal

(iv) Define $f: V(G) \to V(G')$ such that $f(a)=q$, $f(b)=r$, $f(c)=s$, $f(d)=p$

Then f is one-one and onto

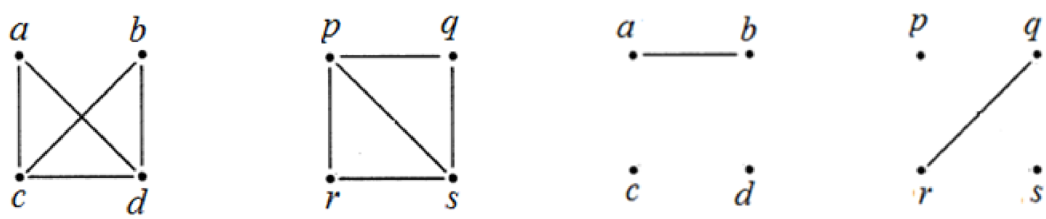
(v) In G and G', for the vertex ordering a, b, c, d and $f(a), f(b), f(c), f(d)$ respectively, the adjacency matrices are given by

$$A_G = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad A_{G^1} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Therefore, $A_G = A_{G^1}$. Hence G and G' are isomorphic

Note: Alternately we can do the above problem using the following.
Two graphs G and G' are isomorphic if and only if the complement graphs of G and G' are isomorphic

Consider the complements of the given graphs G and G'



G $\qquad G'$ $\qquad \overline{G}$ $\qquad \overline{G'}$

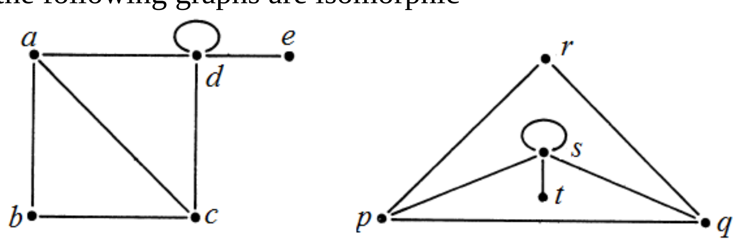
Define $f: V(\overline{G}) \to V(\overline{G'})$ such that $f(a)=q$, $f(b)=r$, $f(c)=s$, $f(d)=p$

Then f is one-one, onto and edge preserving.

Therefore, the graphs $\overline{G}$ and $\overline{G'}$ are isomorphic

Hence the graphs G and G' are isomorphic

Example 29: Show that the following graphs are isomorphic



G $\qquad G'$

Solution: Here

(i) $|V(G)|=5$ and $|V(G')|=5$ $\quad \therefore |V(G)|=|V(G')|$

(ii) $|E(G)|=7$ and $|E(G')|=7$ $\quad \therefore |E(G)|=|E(G')|$

(iii) In G, $\deg(a)=3$, $\deg(b)=2$, $\deg(c)=3$, $\deg(d)=5$, $\deg(e)=1$ and the degree sequence is $(1, 2, 3, 3, 5)$

In G', $\deg(p)=3$, $\deg(q)=3$, $\deg(r)=2$, $\deg(s)=5$, $\deg(t)=1$ and the degree sequence is $(1, 2, 3, 3, 5)$

$\therefore$ The degree sequences are equal

(iv) Define $f:V(G)\to V(G')$ such that $f(e)=t,\ f(b)=r,\ f(d)=s,\ f(a)=p,\ f(c)=q$

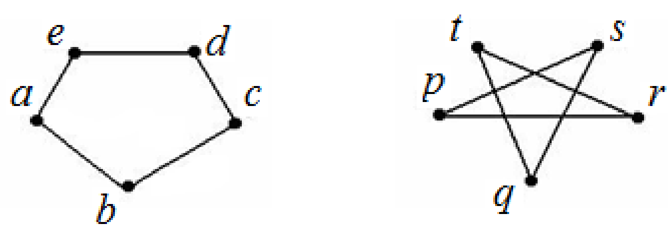
Then f is one-one and onto

(v) In G and G', for the vertex ordering a, b, c, d, e and p, r, q, s, t respectively, the adjacency matrices are given by

$$A_G=\begin{bmatrix}0&1&1&1&0\\1&0&1&0&0\\1&1&0&1&0\\1&0&1&1&1\\0&0&0&1&0\end{bmatrix}\qquad A_{G^1}=\begin{bmatrix}0&1&1&1&0\\1&0&1&0&0\\1&1&0&1&0\\1&0&1&1&1\\0&0&0&1&0\end{bmatrix}$$

Therefore, $A_G=A_{G^1}$. Hence G and G' are isomorphic

Example 30: Show that the following graphs are isomorphic



G $\qquad\qquad$ G'

Solution: Here

(i) $|V(G)|=5$ and $|V(G')|=5$ $\quad\therefore |V(G)|=|V(G')|$

(ii) $|E(G)|=5$ and $|E(G')|=5$ $\quad\therefore |E(G)|=|E(G')|$

(iii) In G, $\deg(a)=2$, $\deg(b)=2$, $\deg(c)=2$, $\deg(d)=2$, $\deg(e)=2$ and the degree sequence is $(2,\ 2,\ 2,\ 2,\ 2)$

In G', $\deg(p)=2$, $\deg(q)=2$, $\deg(r)=2$, $\deg(s)=2$, $\deg(t)=2$ and the degree sequence is $(2,\ 2,\ 2,\ 2,\ 2)$

$\therefore$ The degree sequences are equal

(iv) G is a cycle $a-b-c-d-e-a$ of length 5 and G' is also a cycle $p-r-t-q-s-p$ of length 5

(v) Define $f:V(G)\to V(G')$ such that $f(a)=p,\ f(b)=r,\ f(c)=t,\ f(d)=q,\ f(e)=s$

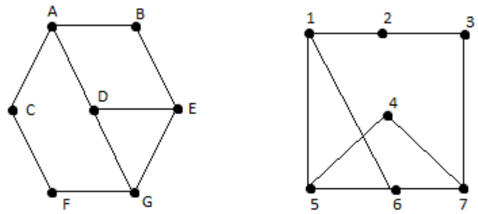
Then f is one-one, onto and edge preserving.

(vi) In G and G', for the vertex ordering a, b, c, d, e and p, r, t, q, s respectively, the adjacency matrices are given by

$$A_G=\begin{bmatrix}0&1&0&0&1\\1&0&1&0&0\\0&1&0&1&0\\0&0&1&0&1\\1&0&0&1&0\end{bmatrix}\qquad A_{G^1}=\begin{bmatrix}0&1&0&0&1\\1&0&1&0&0\\0&1&0&1&0\\0&0&1&0&1\\1&0&0&1&0\end{bmatrix}$$

Therefore, $A_G=A_{G^1}$. Hence G and G' are isomorphic

Example 31: Show that the following graphs are isomorphic



G $\quad$ G'

Solution: Here

(i) $|V(G)|=7$ and $|V(G')|=7$ $\quad \therefore |V(G)|=|V(G')|$

(ii) $|E(G)|=9$ and $|E(G')|=9$ $\quad \therefore |E(G)|=|E(G')|$

(iii) In G, $\deg(A)=3, \deg(B)=2, \deg(C)=2, \deg(D)=3, \deg(E)=3, \deg(F)=2, \deg(G)=3$ and the degree sequence is $(2,\ 2,\ 2,\ 3,\ 3,\ 3,\ 3)$

In G', $\deg(1)=3, \deg(2)=2, \deg(3)=2, \deg(4)=2, \deg(5)=3, \deg(6)=3, \deg(7)=3$ and the degree sequence is $(2,\ 2,\ 2,\ 3,\ 3,\ 3,\ 3)$

$\therefore$ The degree sequences are equal

(iv) Consider the vertices of degree 2 in two graphs B, C, F and $2, 3, 4$

Since C, F are adjacent and $2, 3$ are adjacent, we should have $B \to 4$

Other vertex adjacent to C is A which is adjacent to vertices of degree 2 and 3

In G', other vertex adjacent to 3 is 7 which is adjacent to vertices of degree 2 and 3

So, $C \to 3, A \to 7, F \to 2$

A is adjacent to D which is of degree 3 and in G', 7 is adjacent to 6 which is of degree 3 So, $D \to 6$

F is adjacent to G which is of degree 3 and in G', 2 is adjacent to 1 which is of degree 3 So, $G \to 1$

The remaining vertex $E \to 5$

Define $f: V(G) \to V(G')$ such that

$f(A)=7,\ f(B)=4,\ f(C)=3,\ f(D)=6,\ f(E)=5,\ f(F)=2,\ f(G)=1$

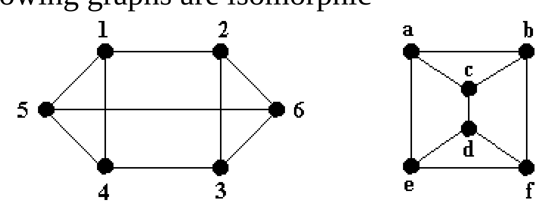
Then f is one-one and onto

(v) In G and G', for the vertex ordering A, B, C, D, E, F, G and $7, 4, 3, 6, 5, 2, 1$ respectively, the adjacency matrices are given by

$A_G =$ $\qquad$ $A_{G^1} =$

Therefore, $A_G = A_{G^1}$. Hence G and G' are isomorphic

Example 32: Show that the following graphs are isomorphic



G $\quad$ G'

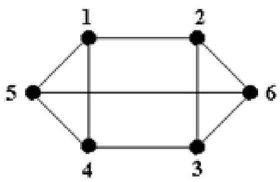
Solution: Here

(i) $|V(G)|=6$ and $|V(G')|=6$ $\quad \therefore |V(G)|=|V(G')|$

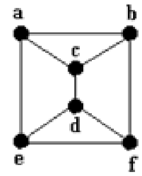
(ii) $|E(G)|=9$ and $|E(G')|=9$ $\quad \therefore |E(G)|=|E(G')|$

(iii) The degree sequence of G and G' are equal to $(3, 3, 3, 3, 3, 3)$

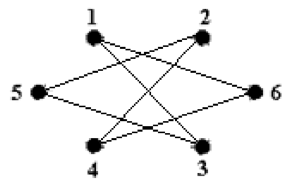
(iv) Consider the complements $\overline{G}$ and $\overline{G'}$ of the graphs G and G' respectively



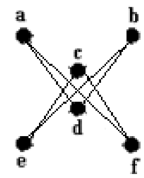
G



G'



$\overline{G}$



$\overline{G'}$

(v) We can observe that $\overline{G}$ is a cycle $1-3-5-2-4-6-1$ of length 6 and $\overline{G'}$ is also a cycle $a-d-b-e-c-f-a$ of length 6

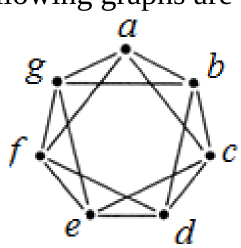
(vi) Define $f: V(\overline{G}) \to V(\overline{G'})$ such that $f(1)=a,\ f(3)=d,\ f(5)=b,\ f(2)=e,\ f(4)=c,\ f(6)=f$

Then f is one-one, onto and edge preserving.

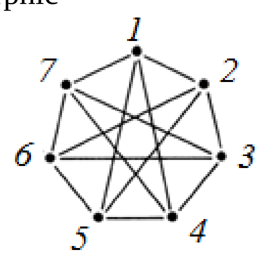
Therefore, the graphs $\overline{G}$ and $\overline{G'}$ are isomorphic

Hence the graphs G and G' are isomorphic

Example 33: Show that the following graphs are isomorphic



G



G'

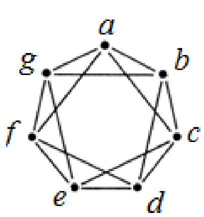
Solution: Here

(i) $|V(G)|=7$ and $|V(G')|=7 \quad \therefore |V(G)|=|V(G')|$

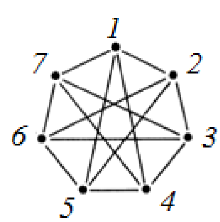
(ii) $|E(G)|=14$ and $|E(G')|=14 \quad \therefore |E(G)|=|E(G')|$

(iii) All the vertices of G and G' are of degree 4

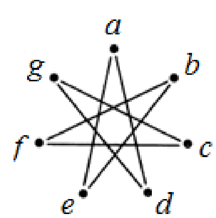
(iv) Consider the complements $\overline{G}$ and $\overline{G'}$ of the graphs G and G' respectively



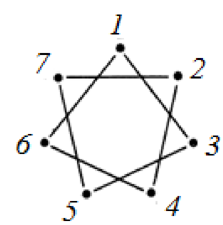
G



G'



$\overline{G}$



$\overline{G'}$

(v) We can observe that $\overline{G}$ is a cycle $a-d-g-c-f-b-e-a$ of length 7 and $\overline{G'}$ is also a cycle $1-3-5-7-2-4-6-1$ of length 7

(vi) Define $f: V(\overline{G}) \to V(\overline{G'})$ such that

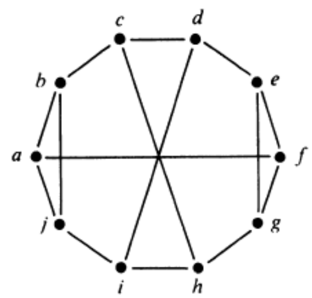
$f(a)=1,\ f(d)=3,\ f(g)=5,\ f(c)=7,\ f(f)=2,\ f(b)=4,\ f(e)=6$

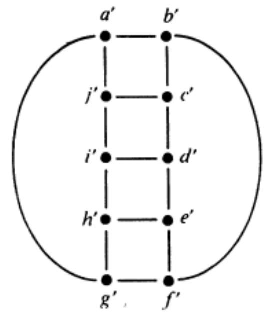
Then f is one-one, onto and edge preserving.

Therefore, the graphs $\overline{G}$ and $\overline{G'}$ are isomorphic

Hence the graphs G and G' are isomorphic

Example 34: Show that the following graphs are not isomorphic





G $\qquad\qquad$ G'

Solution: Here

(i) $|V(G)|=10$ and $|V(G')|=10$ $\quad\therefore |V(G)|=|V(G')|$

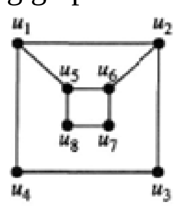
(ii) $|E(G)|=15$ and $|E(G')|=15$ $\quad\therefore |E(G)|=|E(G')|$

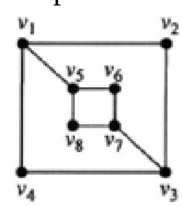
(iii) All the vertices of G and G' are of degree 3

(iv) Observe that the graph G has a cycle of length 3, where as the graph G' has no cycles of length 3

Therefore, G and G' are not isomorphic

Example 35: Show that the following graphs are not isomorphic





G $\qquad\qquad$ H

Solution: Here

(i) $|V(G)|=8$ and $|V(H)|=8$ $\quad\therefore |V(G)|=|V(H)|$

(ii) $|E(G)|=10$ and $|E(H)|=10$ $\quad\therefore |E(G)|=|E(G')|$

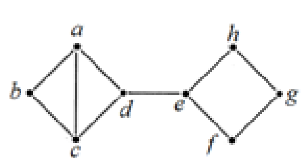
(iii) The degree sequence of G and H are equal to (2, 2, 2, 2, 3, 3, 3, 3)

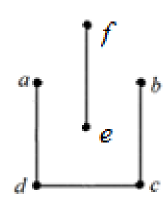
(iv) Observe that in the graph G there are adjacent pairs of vertices $\{u_3, u_4\}$ and $\{u_7, u_8\}$ of degree 2, where as in the graph H there are no adjacent pairs of vertices of degree 2

Therefore, G and H are not isomorphic

Connected graph: A graph G is called *connected* if there is a path between every pair of vertices

Example 36: Consider the following graphs G and G'



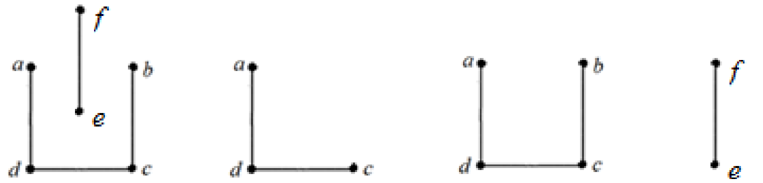


G $\qquad\qquad$ G'

Here in the graph G, there is a path between every pair of vertices. Therefore, G is a connected graph.
In the graph G' there is no path between the vertices a and e. Therefore, G' is not a connected graph

Connected component: A connected subgraph H of a graph G is called *connected component* or *component* if H is not contained in a larger connected subgraph of G

Example 37: Consider the following graphs G and the subgraphs H_1, H_2 and H_3



G $\qquad H_1$ $\qquad H_2$ $\qquad H_3$

Here

(i) G is not a connected graph

(ii) H_1, H_2 and H_3 are connected subgraphs of G

(iii) H_1 is not a connected component (since H_1 is contained in the larger connected subgraph H_2)

(iv) H_2 is a connected component (since it is not contained in any larger connected subgraph)

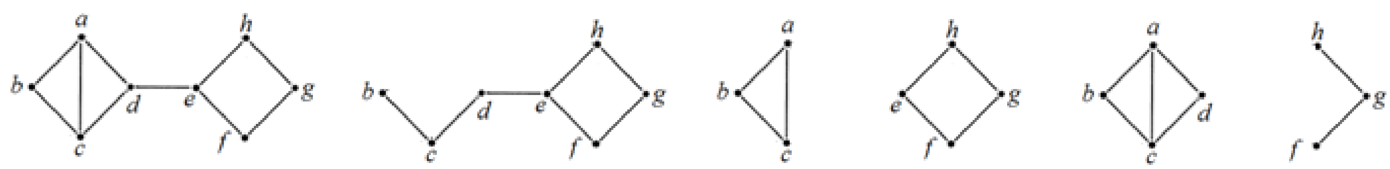
(v) H_3 is a connected component (since it is not contained in any larger connected subgraph)

(vi) H_2 and H_3 are the only connected components (components) of G

Cut vertex: A vertex v of a graph G is called a *cut vertex* if the graph $G-v$ is not connected

Cut edge: An edge e of a graph G is called a *cut edge* or a *bridge* if the graph $G-e$ is not connected

Example 38: Consider the following graphs G



G $\qquad G-a$ $\qquad G-d$ $\qquad G-e$

Here

(i) G is a connected graph

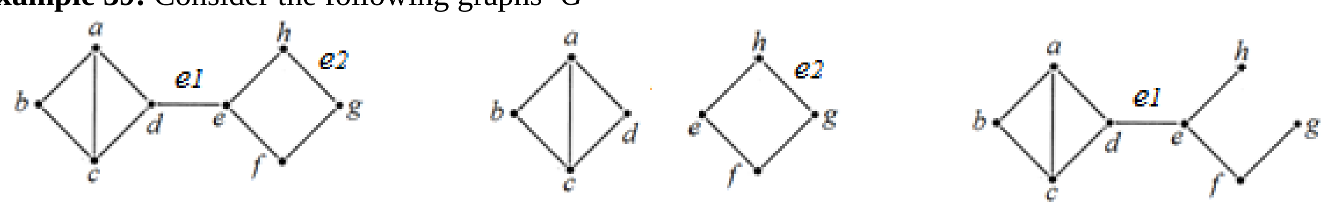
(ii) a is not a cut vertex (since $G-a$ is connected)

(iii) d is a cut vertex (since $G-d$ is not connected)

(iv) e is a cut vertex (since $G-e$ is not connected)

(v) d and e are the only cut vertices

Example 39: Consider the following graphs G



G $\qquad G-e_1$ $\qquad G-e_2$

Here

(i) G is a connected graph

(ii) $G-e_1$ is a cut edge or bridge (since $G-e_1$ is not connected)

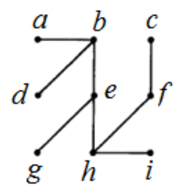
(iii) $G-e_2$ is not a cut edge (since $G-e_2$ is connected)

(iv) e_1 is the only cut edge of G

Tree: A simple graph in which there is a simple unique path between every pair of vertices is called a *tree*. A tree with only one vertex is called a *trivial tree*.
Rooted tree: A tree with a designated vertex as a root is called a *rooted tree*
Level of a vertex: In a rooted tree the length of the path from a vertex v to the root is called the *length of the vertex* v
Note: Let G be graph with n vertices and $n>1$. Then the following are equivalent

(i) G is a tree
(ii) G is connected and has no cycles
(iii) G has no cycles with $n-1$ edges
(iv) G is connected with $n-1$ edges

Example 40: Consider the following graph T



T

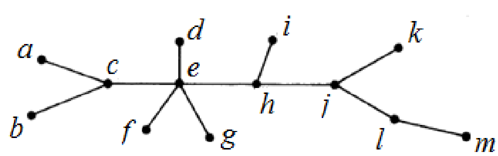
Here
(i) T is connected and has no cycles, so T is a tree
(ii) T has 9 vertices and 8 edges
(iii) If the vertex a is designated as the root, then it becomes a rooted tree. In this case,

Level	1	2	3	4	5
Vertices	b	d,e	g,h	i,f	c

(iv) If the vertex e is designated as the root, then also it becomes a rooted tree. In this case,

Level	1	2	3
Vertices	b,g,h	a,d,i,f	c

Example 41: Consider the following graph T



T

Here
(i) T is connected and has no cycles, so T is a tree
(ii) T has 13 vertices and 12 edges
(iii) If the vertex c is designated as the root, then it becomes a rooted tree. In this case,

Level	1	2	3	4	5
Vertices	a,b,e	d,f,g,h	i,j	k,l	m

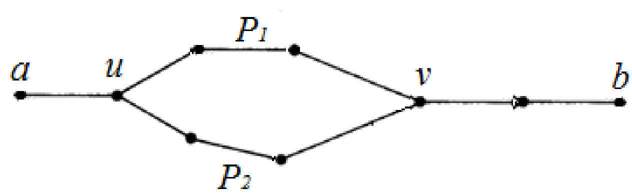
(iv) If the vertex k is designated as the root, then also it becomes a rooted tree. In this case,

Level	1	2	3	4	5
Vertices	j	h,l	e,i,m	c,d,f,g	a,b

Theorem 5: A simple non directed graph G is a tree if and only if G is connected and has no cycles
Proof: Suppose that G is a tree. Then by the definition of tree there is a simple unique path between every pair of vertices. Therefore, G is connected.
If G contains a cycle, then any two vertices on this cycle have at least two distinct simple paths which leads to a contradiction. Therefore G has no cycles.

Conversely suppose that G is connected and has no cycles.
Then there is a path between every pair of vertices.

Suppose that there are two simple paths P_1 and P_2 between a pair of vertices a and b
Since P_1 and P_2 are different paths, there exists a vertex u (say) on both the paths such that the vertex following u on P_1 is different from the vertex following u on P_2
Since P_1 and P_2 terminates at b, there exists first vertex after u say v which is a common vertex on both P_1 and P_2.



Therefore, the $u-v$ path along P_1 together with the $v-u$ path along P_2 forms a cycle in G, which contradicts that G has no cycles.
Therefore there is a simple unique path between every pair of vertices and hence G is a tree

Theorem 6: Every non trivial finite tree contains at least one vertex of degree one
Prof: Let T be a non trivial finite tree. Choose a vertex v_1 in T
If $\deg(v_1)=1$, then the theorem is proved. Otherwise choose a vertex v_2 other than v_1 and adjacent to v_1
If $\deg(v_2)=1$, then the theorem is proved. Otherwise choose a vertex v_3 other than v_2 and adjacent to v_2
Continuing this process, we have a path of the form $v_1-v_2-v_3-v_4-\cdots$
In this path no vertex is repeated as T does not contain cycles.
Now since T is finite, this path is terminated at a vertex say v_k
Then obviously $\deg(v_k)=1$
Therefore, T contains at least one vertex of degree one

Theorem 7: A tree with n vertices has exactly $n-1$ edges
Prof: Let T be a tree with n vertices.
We prove this theorem by applying mathematical induction on n
If $n=1$, then the tree contains only one vertex but no edges and therefore the theorem is proved
Assume that the theorem is true for $n=k$; that is tree with k vertices has exactly $k-1$ edges
Now we prove the theorem for $n=k+1$
Suppose that T has $k+1$ vertices. Then a known theorem T has at least one vertex v of degree 1
Now consider the graph $T-v$
Then $T-v$ is a tree with k vertices and one edge less than that of T
Therefore, by the induction assumption $T-v$ has exactly $k-1$ edges and hence T has exactly k edges
Thus the theorem is proved for $n=k+1$
Therefore, every tree with n vertices has exactly $n-1$ edges

Theorem 8: Every non trivial tree contains at least two vertices of degree one

Prof: Let T be a non trivial tree with n vertices. Let $V(T)=\{v_1, v_2, \cdots v_n\}$.
Then T has $n-1$ edges and by the sum of degrees theorem, we have

$$\sum_{i=1}^{n} \deg(v_i) = 2|E| = 2(n-1) = 2n-2 \quad \cdots \quad \cdots \quad \cdots \quad (i)$$

Suppose that T has only one vertex say v_1 of degree 1, then $\deg(v_i) \geq 2$ for $i = 2,3,4, \cdots n$

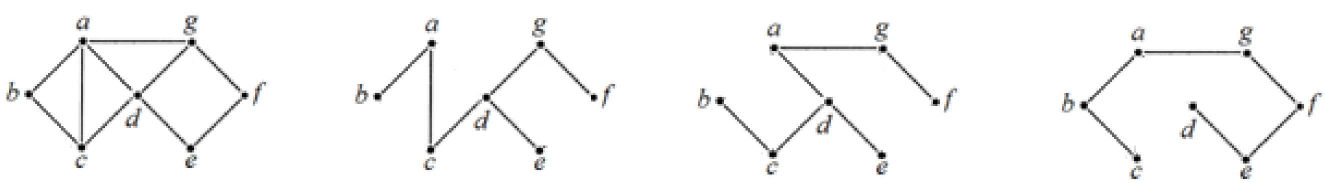
Consider, $\sum_{i=2}^{n} \deg(v_i) = \deg(v_2) + \deg(v_3) + \cdots + \deg(v_n)$

$$\begin{aligned} &\geq 2+2+\cdots (n-1) \text{ times} \\ &= 2(n-1) \\ &= \sum_{i=1}^{n} \deg(v_i) \\ &= 1 + \sum_{i=2}^{n} \deg(v_i) \quad \because \deg(v_1) = 1 \end{aligned}$$

This implies that $0 \geq 1$, a contradiction
Therefore, T contains at least two vertices of degree one

Spanning tree: A subgraph H of a graph G is called a spanning tree of G if H is a tree and contains all the vertices of G
Note: A graph G is connected if and only if G contains a spanning tree

Example 42: Consider the following graph G and spanning trees H_1, H_2 and H_3



G H_1 H_2 H_3

BFS and DFS algorithms: There are two algorithms for finding a spanning tree from a connected graph are known as Breadth-first search (BFS) and Depth-first search (DFS)
Note: The idea of BFS is to visit all the vertices sequentially on a given level before going to the next level where as the idea of DFS proceeds successively to higher levels at the first opportunity

BFS algorithm for a Spanning Tree:

Algorithm 5.4.1. Breadth-First Search for a Spanning Tree.

Input: A connected graph G with vertices labeled $v_1, v_2, \ldots, v_n$.
Output: A spanning tree T for G.
Method:

1. (Start) Let v_1 be the root of T. Form the set $V = \{v_1\}$.
2. (Add new edges.) Consider the vertices of V in order consistent with the original labeling. Then for each vertex $x \in V$, add the edge $\{x, v_k\}$ to T where k is the minimum index such that adding the edge $\{x, v_k\}$ to T does not produce a cycle. If no edge can be added, then stop; T is a spanning tree for G. After all the vertices of V have been considered in order, go to Step 3.
3. (Update V.) Replace V by all the children v in T of the vertices x of V where the edges $\{x, v\}$ were added in Step 2. Go back and repeat Step 2 for the new set V.

Algorithm 8.6 (Breadth-first Search): This algorithm executes a breadth-first search on a graph G beginning with a starting vertex A.

Step 1. Initialize all vertices to the ready state (STATUS = 1).

Step 2. Put the starting vertex A in QUEUE and change the status of A to the waiting state (STATUS = 2).

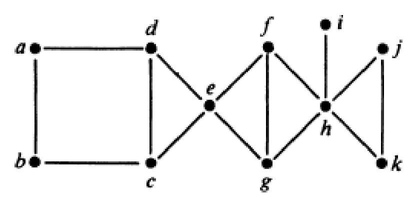
Step 3. Repeat Steps 4 and 5 until QUEUE is empty.

Step 4. Remove the front vertex N of QUEUE. Process N, and set STATUS (N) = 3, the processed state.

Step 5. Examine each neighbor J of N.
(*a*) If STATUS (J) = 1 (ready state), add J to the rear of QUEUE and reset STATUS (J) = 2 (waiting state).
(*b*) If STATUS (J) = 2 (waiting state) or STATUS (J) = 3 (processed state), ignore the vertex J.
[End of Step 3 loop.]

Step 6. Exit.

Example 42: Using BFS algorithm, determine a spanning tree from the following graph



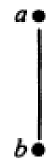
Solution: Let T be the required spanning tree or BFS tree

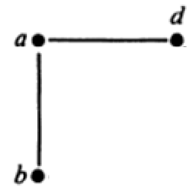
(i) Fix the order of the vertices as $a, b, c, d, e, f, g, h, i, j, k$

(ii) **Step 1:** Fix the vertex a as root
At this stage T contains only one element, $T = \{a\}$

(iii) **Step 2:** Add to T all edges $\{a, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.
We add the edges first $\{a, b\}$ and then $\{a, d\}$. These two edges are tree edges of the BFS tree.

At this stage $T = \{a, b, d\}$ with b, d are at level 1 vertices





(iv) **Step 3: Repeat the step 2** for all the level 1 vertices b, d in the specified vertex order

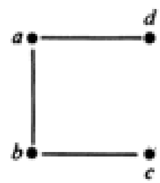
Add to T all edges $\{b, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

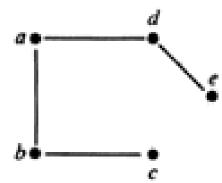
We add the edge $\{b, c\}$ as a BFS tree edge

Now add to T all edges $\{d, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

We add the edge $\{d, e\}$ as a BFS tree edge

At this stage $T = \{a, b, d, c, e\}$ with b, d are at level 1 vertices and c, e are at level 2 vertices





(v) **Repeat the step 2** for all the level 2 vertices c, e in the specified vertex order

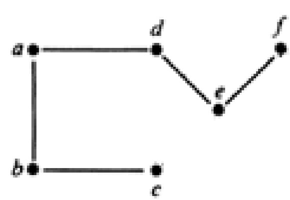
Add to T all edges $\{c, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

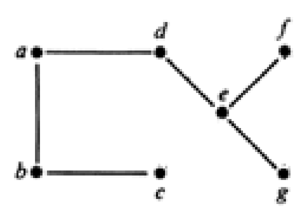
No edge is added as a BFS tree edge

Now add to T all edges $\{e, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

We add the edges first $\{e, f\}$ and then $\{e, g\}$ as BFS tree edges.

At this stage $T = \{a, b, d, c, e, f, g\}$ with b, d are at level 1 vertices; c, e are at level 2 vertices and f, g are at level 3 vertices





(vi) **Repeat the step 2** for all the level 3 vertices f, g in the specified vertex order

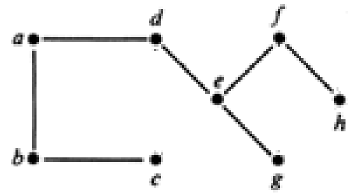
Add to T all edges $\{f, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

We add the edge $\{f, h\}$ as a BFS tree edge

Now add to T all edges $\{g, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

No edge is added as a BFS tree edge

At this stage $T = \{a, b, d, c, e, f, g, h\}$ with b, d are at level 1 vertices; c, e are at level 2 vertices; f, g are at level 3 vertices and h is at level 4 vertex

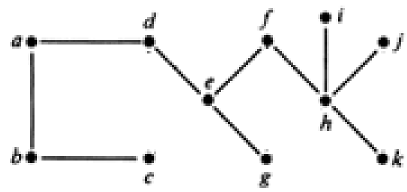


(vii) **Repeat the step 2** for the level 4 vertex h in the specified vertex order

Now add to T all edges $\{h, x\}$ such that each x is not in T, x in the specified vertex order and addition of an edge does not produce a cycle in T.

We add the edges first $\{h, i\}$ then $\{h, j\}$ and then $\{h, k\}$ as BFS tree edges.

At this stage $T = \{a, b, d, c, e, f, g, h, i, j, k\}$ with b, d are at level 1 vertices; c, e are at level 2 vertices; f, g are at level 3 vertices; h is at level 4 vertex and i, j, k are at level 5 vertices



Since T contains all the vertices of the given graph, the above graph is the required BFS spanning tree

Note:

1	Status 1: Ready State	$a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	-
	Status 3: Processed State	-
2	Status 1: Ready State	$b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	a
	Status 3: Processed State	
3	Status 1: Ready State	$c,\ e,\ f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	$b,\ d$
	Status 3: Processed State	a
4	Status 1: Ready State	$e,\ f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	$d\ ,\ c$
	Status 3: Processed State	$a\ ,\ b$
5	Status 1: Ready State	$f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	$c\ ,\ e$
	Status 3: Processed State	$a\ ,\ b\ ,d$
6	Status 1: Ready State	$f,\ g,\ h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	e
	Status 3: Processed State	$a\ ,\ b\ ,d\ ,c$
7	Status 1: Ready State	$h,\ i,\ j,\ k$
	Status 2: Waiting state (Queue)	$f\ ,\ g$
	Status 3: Processed State	$a\ ,\ b\ ,d\ ,c\ ,e$
8	Status 1: Ready State	$i,\ j,\ k$
	Status 2: Waiting state (Queue)	$g\ ,\ h$

	Status 3: Processed State	a, b, d, c, e, f
9	Status 1: Ready State	i, j, k
	Status 2: Waiting state (Queue)	h
	Status 3: Processed State	a, b, d, c, e, f, g
10	Status 1: Ready State	
	Status 2: Waiting state (Queue)	i, j, k
	Status 3: Processed State	a, b, d, c, e, f, g, h
11	Status 1: Ready State	
	Status 2: Waiting state (Queue)	
	Status 3: Processed State	a, b, d, c, e, f, g, h, i, j, k

DFS algorithm for a Spanning Tree:

Algorithm 5.4.2. Depth-First Search for a Spanning Tree.

Input: A connected graph G with vertices labeled $v_1, v_2, \ldots, v_n$.
Output: A spanning tree T for G.
Method:

1. (Visit a vertex.) Let v_1 be the root of T, and set $L = v_1$. (The name L stands for the vertex last visited.)
2. (Find an unexamined edge and an unvisited vertex adjacent to L.) For all vertices adjacent to L, choose the edge $\{L, v_k\}$, where k is the minimum index such that adding $\{L, v_k\}$ to T does not create a cycle. If no such edge exists, go to Step 3; otherwise, add edge $\{L, v_k\}$ to T and set $L = v_k$; repeat Step 2 at the new value for L.
3. (Backtrack or terminate.) If x is the parent of L in T, set $L = x$ and apply Step 2 at the new value of L. If, on the other hand, L has no parent in T (so that $L = v_1$) then the depth-first search terminates and T is a spanning tree for G.

Algorithm 8.5 (Depth-first Search): This algorithm executes a depth-first search on a graph G beginning with a starting vertex A.

Step 1. Initialize all vertices to the ready state (STATUS = 1).

Step 2. Push the starting vertex A onto STACK and change the status of A to the waiting state (STATUS = 2).

Step 3. Repeat Steps 4 and 5 until STACK is empty.

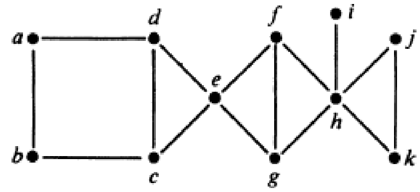
Step 4. Pop the top vertex N of STACK. Process N, and set STATUS (N) = 3, the processed state.

Step 5. Examine each neighbor J of N.

(*a*) If STATUS (J) = 1 (ready state), push J onto STACK and reset STATUS (J) = 2 (waiting state).
(*b*) If STATUS (J) = 2 (waiting state), delete the previous J from the STACK and push the current J onto STACK.
(*c*) If STATUS (J) = 3 (processed state), ignore the vertex J.
[End of Step 3 loop.]

Step 6. Exit.

Example 43: Using DFS algorithm, determine a spanning tree from the following graph



Solution: Let T be the required spanning tree or DFS tree

(i) Fix the order of the vertices as $a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j,\ k$

(ii) **Step 1:** Fix the vertex a as root. Here a is called visited vertex

At this stage T contains only one element, $T=\{a\}$

(iii) **Step 2:** Add to T an edge $\{a,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{a,b\}$ as DFS tree edge. Here a is called the parent of b and b is called the child of a. Also b becomes next visited vertex.

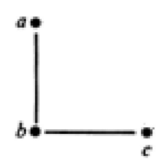
At this stage $T=\{a,b\}$

(iv) **Step 3:** Repeat step 2 with last visited vertex b

Add to T an edge $\{b,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{b,c\}$ as DFS tree edge so that c is the next visited vertex

At this stage $T=\{a,b,c\}$



(v) Repeat step 2 with last visited vertex c

Add to T an edge $\{c,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{c,d\}$ as DFS tree edge so that d is the next visited vertex

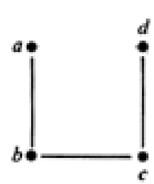
At this stage $T=\{a,b,c,d\}$

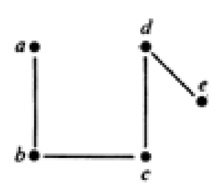
(vi) Repeat step 2 with last visited vertex d

Add to T an edge $\{d,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{d,e\}$ as DFS tree edge so that e is the next visited vertex

At this stage $T=\{a,b,c,d,e\}$





(vii) Repeat step 2 with last visited vertex e

Add to T an edge $\{e,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{e, f\}$ as DFS tree edge so that f is the next visited vertex

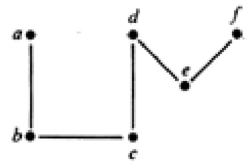
At this stage $T = \{a, b, c, d, e, f\}$

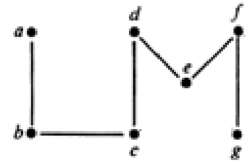
(viii) Repeat step 2 with last visited vertex f

Add to T an edge $\{f, x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{f, g\}$ as DFS tree edge so that g is the next visited vertex

At this stage $T = \{a, b, c, d, e, f, g\}$



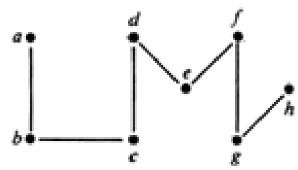


(ix) Repeat step 2 with last visited vertex g

Add to T an edge $\{g, x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{g, h\}$ as DFS tree edge so that h is the next visited vertex

At this stage $T = \{a, b, c, d, e, f, g, h\}$

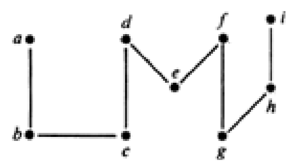


(x) Repeat step 2 with last visited vertex h

Add to T an edge $\{h, x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{h, i\}$ as DFS tree edge so that i is the next visited vertex

At this stage $T = \{a, b, c, d, e, f, g, h, i\}$



(xi) Repeat step 2 with last visited vertex i

Add to T an edge $\{i, x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

No edge can be added as DFS tree edge and in this case go to Step 4.

Step 4: Go back to the parent vertex h of i. Here the process of returning to the parent vertex is called back tracking.

Now repeat step 2 with parent vertex h

Add to T an edge $\{h, x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

We add the edge $\{h, j\}$ as DFS tree edge so that j is the next visited vertex

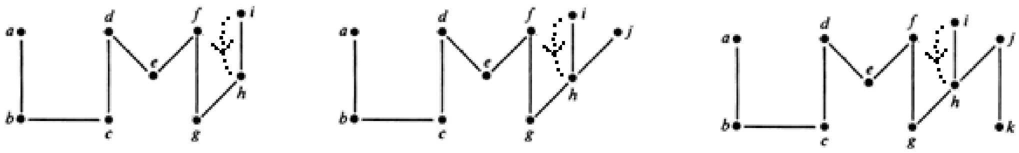
At this stage $T = \{a, b, c, d, e, f, g, h, i, j\}$

(xii) Repeat step 2 with last visited vertex j

Add to T an edge $\{j,x\}$ such that x is not in T, x is the first vertex in the specified vertex order and the addition of this edge does not produce a cycle in T.

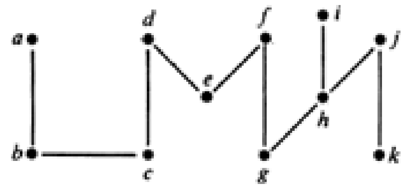
We add the edge $\{j,k\}$ as DFS tree edge so that k is the next visited vertex

At this stage $T=\{a,b,c,d,e,f,g,h,i,k\}$

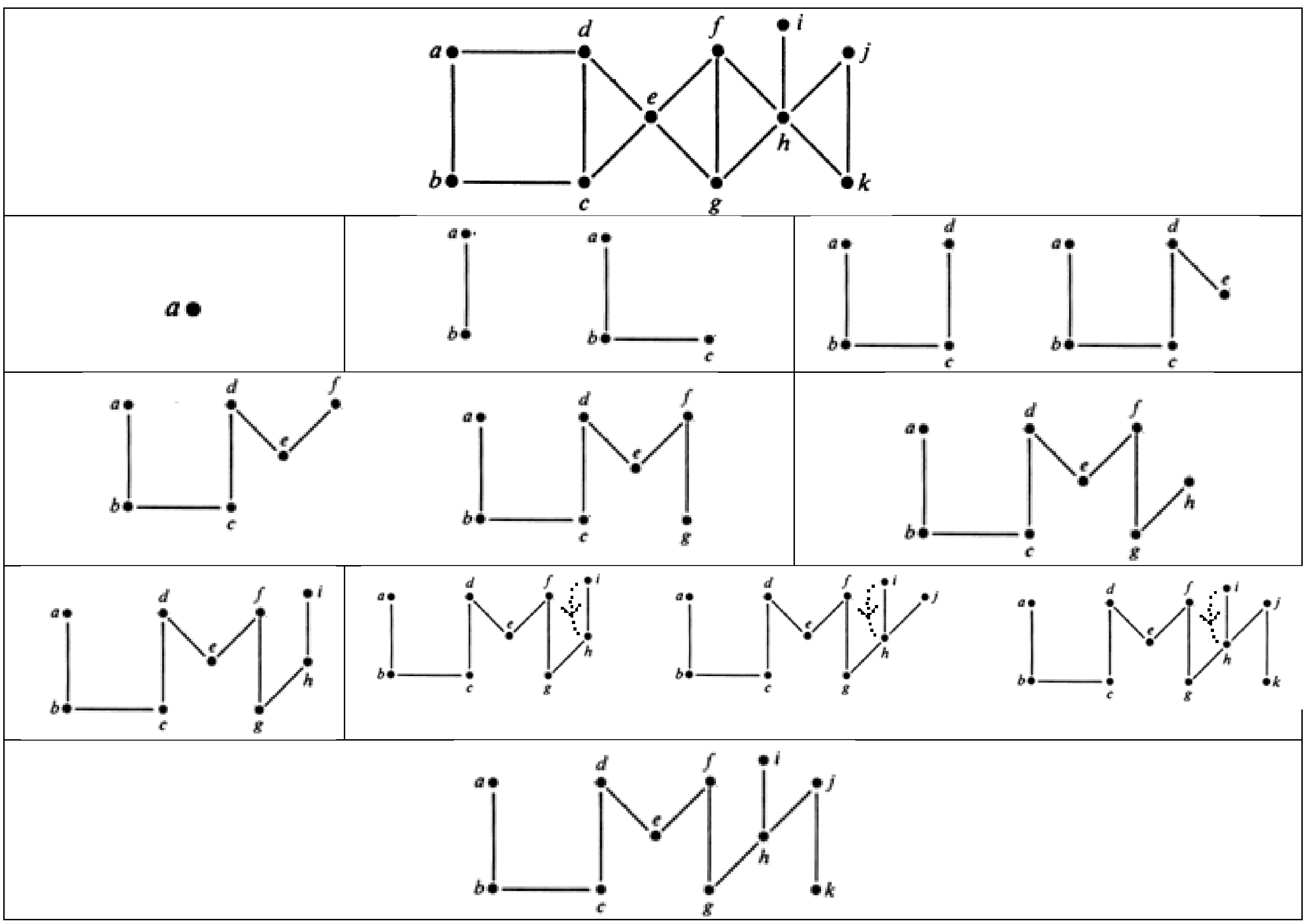


Here the dotted line represents back tracking.

Since T contains all the vertices of the given graph, the above graph is the required BFS spanning tree



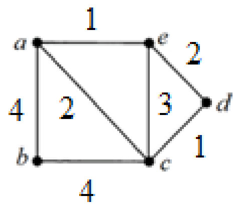
Note:



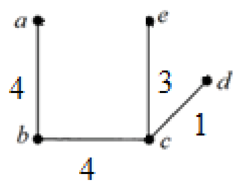
Weighted Graph: A graph in which every edge is assigned a non negative value or a weight is called a weighted graph

Minimal Spanning Tree: A spanning tree T of a weighted graph G is called a minimal spanning tree of G if the sum of all the weights of the edges of T is minimal.

Example 44: Consider the weighted graph G and spanning trees H_1, H_2 and H_3

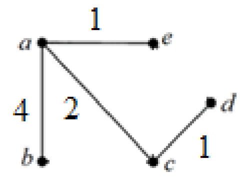


G



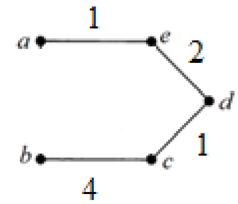
H_1

Total weight: 12



H_2

Total weight: 8



H_3

Total weight: 8

Kruskal's Algorithm:

Algorithm 5.4.3. Kruskal's Algorithm for Finding a Minimal Spanning Tree.

Input: A connected graph G with nonnegative values assigned to each edge.

Output: A minimal spanning tree for G.

Method:

1. Select any edge of minimal value that is not a loop. This is the first edge of T. (If there is more than one edge of minimal value, arbitrarily choose one of these edges.)
2. Select any remaining edge of G having minimal value that does not form a circuit with the edges already included in T.
3. Continue Step 2 until T contains n-1 edges, where n is the number of vertices of G.

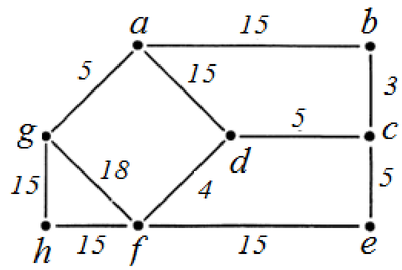
Algorithm 8.3 (Kruskal): The input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in order of increasing weights.

Step 2. Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until $n - 1$ edges are added.

Step 3. Exit.

Example 45: Using Kruskal's algorithm, determine a minimal spanning tree from the following graph

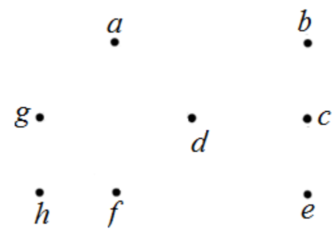


Solution: Let T be the required minimal spanning tree (MST).
Since the given graph contains 8 vertices, T contains 7 edges

(i) Consider the edges of the given graph with increasing order of their weights

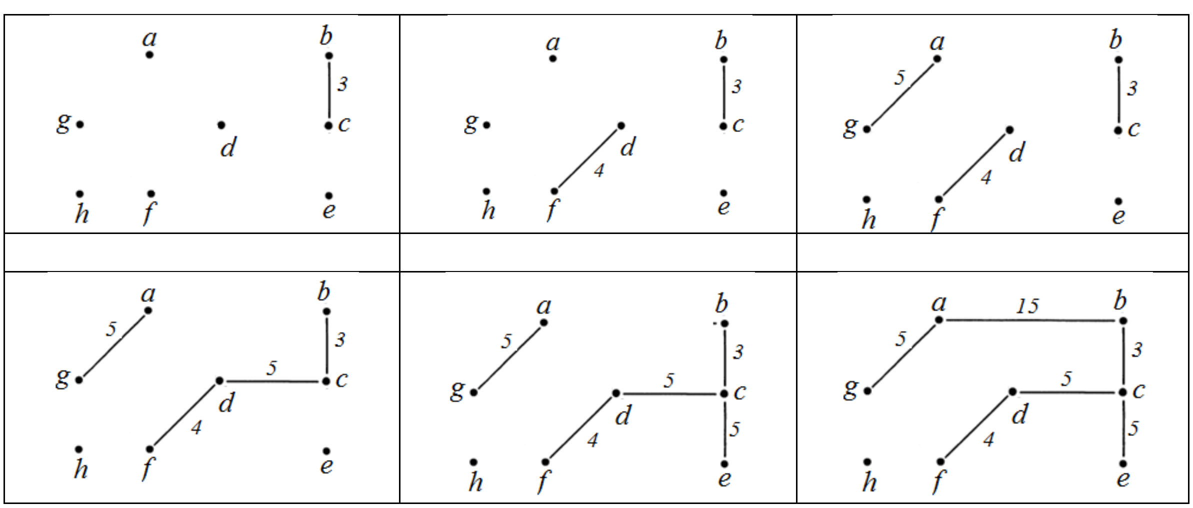
Edge	$\{b,c\}$	$\{d,f\}$	$\{a,g\}$	$\{c,d\}$	$\{c,e\}$	$\{a,b\}$	$\{a,d\}$	$\{e,f\}$	$\{f,h\}$	$\{g,h\}$	$\{f,g\}$
Weight	3	4	5	5	5	15	15	15	15	15	18

(ii) Step 1: Start with the null graph formed by the vertices of the given graph.
At this stage T has no edges and is as follows.



(iii) Step 2: Add edges sequentially in increasing order of their weights and addition of an edge does not produce a cycle. Repeat the Step 2 until we add 7 edges.

We add the edges $\{b,c\}$, $\{d,f\}$, $\{a,g\}$, $\{c,d\}$, $\{c,e\}$, $\{a,b\}$ to T.
At this stage T has 6 edges and is modified as follows.

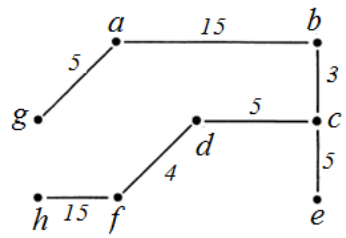


(iv) If we add an edge $\{a,d\}$ or $\{e, f\}$, then it produce a cycle in the required MST and hence go to next edge $\{f,h\}$

Now we add edge $\{f,h\}$ to T and with the addition of this edge T has 7 edges, the required number of edges.

Stop the process.

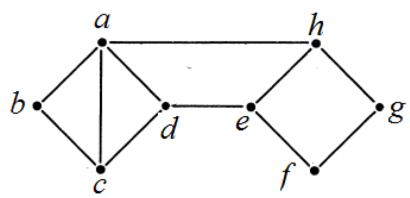
Finally T has 7 edges and is given as follows.



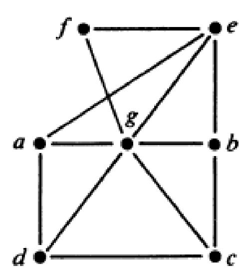
The total weight of the MST T is $3+4+5+5+5+15+15=52$

Exercise:

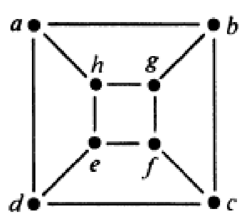
1. Determine a spanning tree from the following graph by using (i) BFS algorithm (ii) DFS algorithm



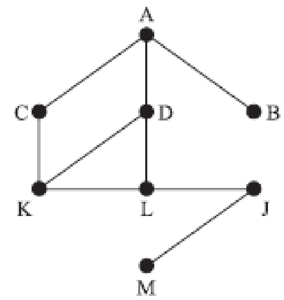
2. Determine a spanning tree from the following graph by using (i) BFS algorithm (ii) DFS algorithm



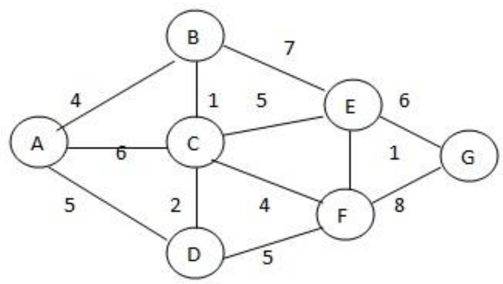
3. Determine a spanning tree from the following graph by using (i) BFS algorithm (ii) DFS algorithm



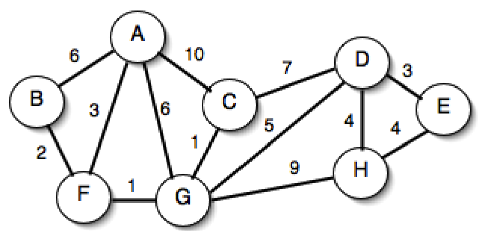
4. Determine a spanning tree from the following graph by using (i) BFS algorithm (ii) DFS algorithm



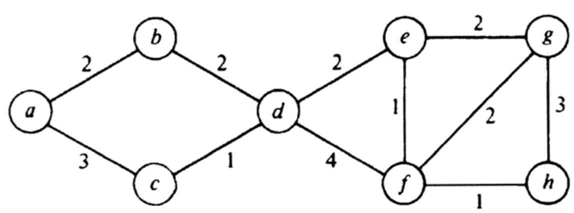
5. Using Kruskal’s algorithm, determine a minimal spanning tree from the following graph



6. Using Kruskal’s algorithm, determine a minimal spanning tree from the following graph



7. Using Kruskal’s algorithm, determine a minimal spanning tree from the following graph



8. Using Kruskal’s algorithm, determine a minimal spanning tree from the following graph