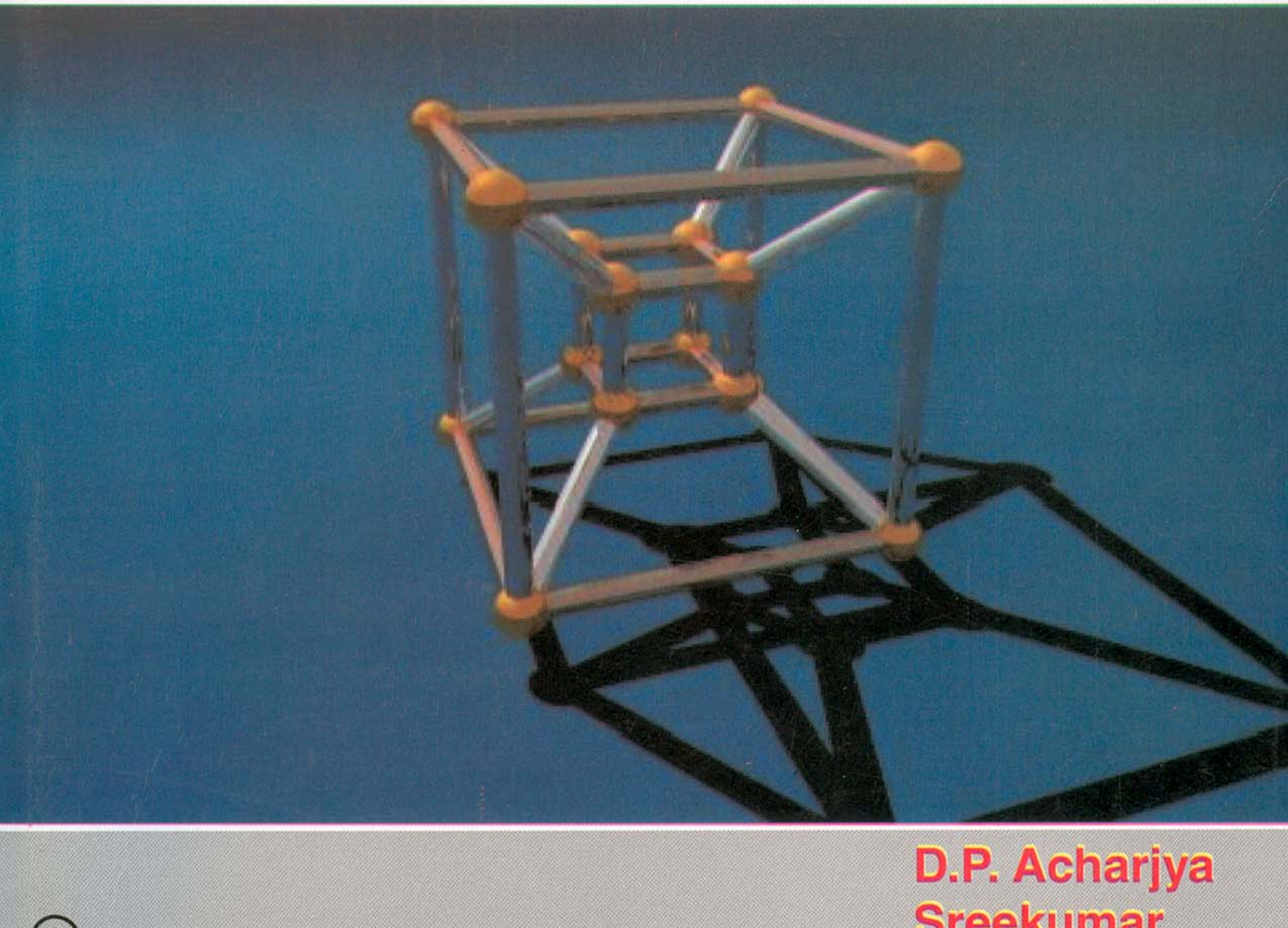
NEW AGE

SECOND EDITION

# FUNDAMENTAL APPROACH TO DISCRETE MATHEMATICS



D.P. Acharjya
Sreekumar

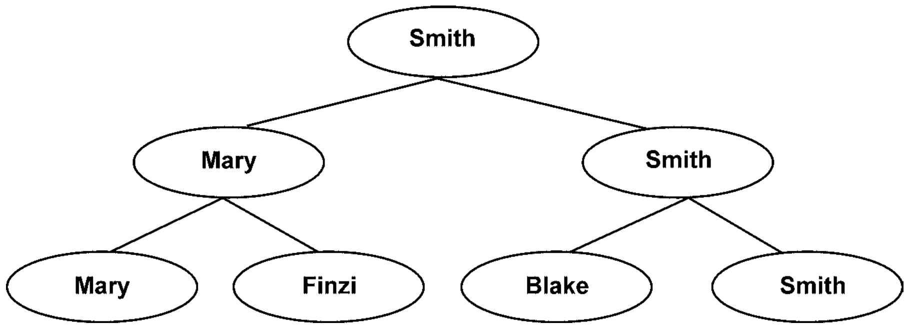
NEW AGE INTERNATIONAL PUBLISHERS

# 13

# Tree

## ■ 13.0 INTRODUCTION

Another very simple and important graph is tree. Computer science makes extensive use of trees. Trees are useful in organizing and relating data in a database. These are interesting not only for their applications to computer science but also for their theoretical properties. Let us consider a single elimination tournament, which means when a player loses, he/she is out of the tournament. Winners continue to play until only one person, the champion, remains. The following figure shows that Mary defeated Finzi and Smith defeated Blake. The winners Mary and Smith, then played, and Smith defeated Mary. *i.e.,* Smith became champion. This is nothing but a tree.



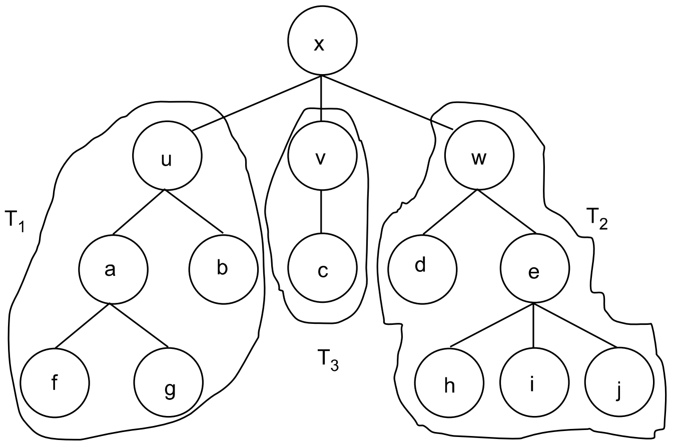
## ■ 13.1 TREE

A connected acyclic graph G is called a tree. A tree T is a finite set of one or more nodes such that

(*i*) There is a specially designated node called the root.

(*ii*) Remaining nodes are partitioned into $k$ disjoint sets $T_1, T_2, ...., T_k$ ; $k > 0$. Where each $T_i$ for $i = 1, 2, ... , k$ is a tree. $T_1, T_2, ... , T_k$ are called subtrees of the root.

In the example given below, the tree T has 14 number of nodes. Which are partitioned into three sets $T_1$, $T_2$ and $T_3$ called the subtrees of T.



## ■ 13.2 FUNDAMENTAL TERMINOLOGIES

A tree has the following fundamental terminologies:

### 13.2.1 Node

The main component of a tree is the node. This stores the actual data and links to the other node.

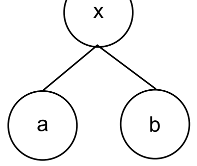
### 13.2.2 Child

Child of a node is the immediate successor of a node. Child, which is at the left side, is called left child and the child, which is at the right side, is called right child.

### 13.2.3 Parent

Parent of a node is the immediate predecessor of a node.

In the figure given below '*x*' is the parent of '*a*' and '*b*'. Where, '*a*' is the left child and '*b*' is the right child of '*x*'.



### 13.2.4 Root

A node that has no parent is termed as the root of the tree. In the above figure '*x*' is termed as the root.

### 13.2.5 Leaf

The node which is at the end and which does not have any child is called leaf node. Leaf node is also termed as terminal node and external node.

### 13.2.6 Level

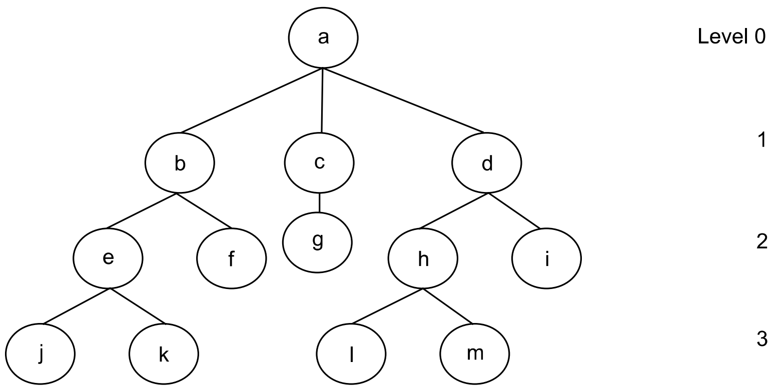
The rank of the hierarchy is known as level. The root node is termed as level 0. If a node is at level $n$, then its parent is at level $(n - 1)$ and child is at level $(n + 1)$.

### 13.2.7 Height

The height $h$ of a tree T is defined as maximum number of nodes that is present in a path starting from root node to a leaf node. The height of a tree is also termed as depth of tree. Mathematically,

$$h = \text{Maximum level} + 1$$

Consider the example of a tree



Height of the tree = Maximum level + 1 = 3 + 1 = 4.

### 13.2.8 Sibling

The nodes, which have the same parent, are termed as siblings. In the above figure $h$ and $i$ are siblings. Similarly $l$ and $m$ are siblings.

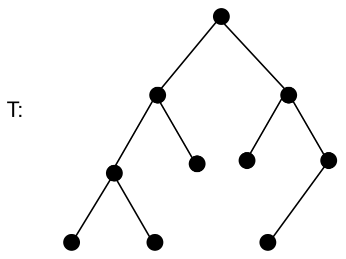
## 13.3 BINARY TREE

A binary tree T is a finite set of nodes such that

(*i*) T is empty or

(*ii*) T contains a specially designed node called the root of T and the remaining nodes of T form two disjoint binary trees $T_1$ and $T_2$. This implies that in case of a binary tree a node may have at most two children.

Consider the following simple binary tree T as



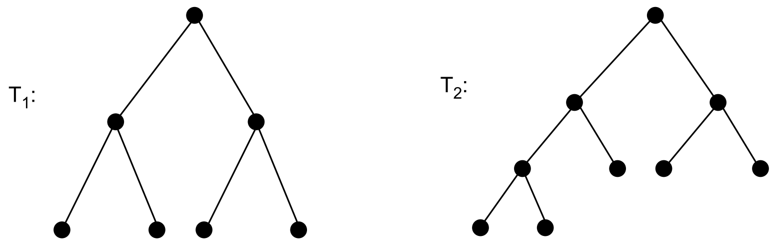
### 13.3.1 Full Binary Tree

A binary tree T is said to be a full binary tree if it contains maximum possible number of nodes in all level. This indicates that, for the level '$n$' of the tree it must contain $2^n$ number of nodes.

### 13.3.2 Complete Binary Tree

A binary tree T is said to be a complete binary tree if it contains maximum possible number of nodes in all levels except the last level.

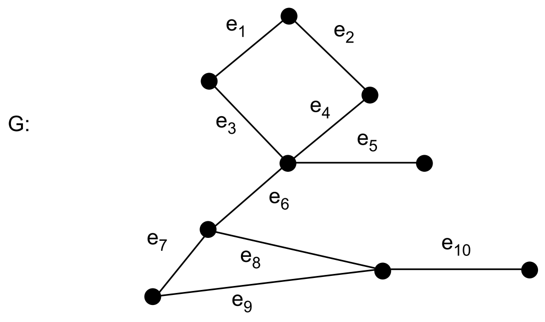
Consider the following examples. In the figure given below $T_1$ is a full binary tree whereas $T_2$ is a complete binary tree.



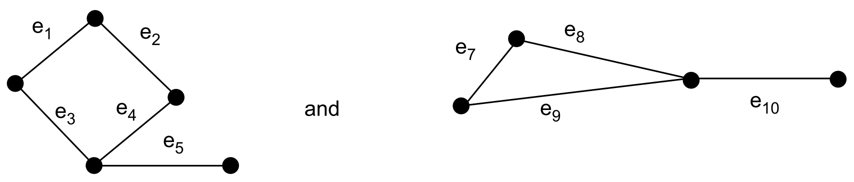
(Full Binary Tree) (Complete Binary Tree)

## ■ 13.4 BRIDGE

An edge of a graph G (V, E) is said to be a bridge if we remove the edge from the graph G, then the graph G has more connected components. Consider the graph G as



On removing the edge $e_6$ from the above graph G, the graph has two connected components such as



Hence, the edge $e_6$ is called as a bridge. In the above figure $e_5$ and $e_{10}$ are also bridges. The bridge is also known as cut edge.

### 13.4.1 Theorem

A tree of order $n$ has size $(n - 1)$.

**Proof:** We prove this by the method of induction.

For $n = 1$ we have a single vertex and hence size is 0.

For $n = 2$, the tree T contains two vertices, so size is 1.

Hence the result follows for $n = 1$ and 2. Assume that the result is true for all trees of order less than $k$. Let T be a tree of order $n = k$ and size $q$, and let $e$ be an edge of T.

We have already observed that $e$ is a bridge of T, so that $(T - e)$ is a forest. Let the two components of $(T - e)$ are $T_1$ and $T_2$, where $T_i$ is a tree of order $n_i$ and size $q_i$ for $i = 1$ and 2.

As, $n_i < k$ for $i = 1$ and 2 so we have $q_1 = (n_1 - 1)$ and $q_2 = (n_2 - 1)$. Since, $n = (n_1 + n_2)$ and $q = (q_1 + q_2 + 1)$ we get

$$q = (n_1 - 1) + (n_2 - 1) + 1 = (n_1 + n_2) - 1 = (n - 1)$$

Therefore, by induction the size of a tree is $(n - 1)$, *i.e.,* one less than its order.

### 13.4.2 Theorem

Every non-trivial tree contains at least two end vertices.

**Proof:** Suppose that T be a tree of order $n$ and size $q$. Let $d_1, d_2, \ldots, d_n$ denote the degrees of its vertices, ordered so that $d_1 \le d_2 \le d_3 \le \ldots \le d_n$. Since T is connected and non-trivial, $d_i \ge 1$ for each $i$; $1 \le i \le n$.

Assume that T does not contain two end-vertices. Hence $d_1 \ge 1$ and $d_i \ge 2$ for $2 \le i \le n$. Thus,

$$\sum_{i=1}^{n} d_i = d_1 + d_2 + d_3 + \ldots. + d_n \ge 1 + 2(n - 1) = 2n - 1 \qquad \ldots (i)$$

But we know $$\sum_{i=1}^{n} d_i = 2q = 2\,(n - 1) = 2n - 2$$

This contradicts inequality $(i)$. So our assumption is wrong. Hence, T contains at least two end-vertices.

## ■ 13.5 DISTANCE AND ECCENTRICITY

In general, a graph's edges are stretchable. As a result, we cannot measure distances in a graph with linear measures. When we look at a graph, however, we can still sense that some parts of the graph are further apart than others. So, in order to quantity a graph the concept of distance and eccentricity is developed.

Let $u$ and $v$ be two vertices of the graph G. The distance between $u$ and $v$ is denoted by $d(u, v)$ and is defined as the length of a shortest $u - v$ path. If there is no path between $u$ and $v$, then $d(u, v) = \infty$.

Let V be the vertex set of G. Let $v \in V$. The eccentricity of $v$ is denoted as $e(v)$ and is defined as

$$e(v) = \text{Max}\,\{d\,(u, v) : u \in V \text{ and } u \ne v\}$$

### 13.5.1 Radius and Diameter

Let G be the graph, then the radius of G is denoted as rad (G) and is defined as

$$\text{rad}(G) = \text{Min}\{e(v): v \in V\}.$$

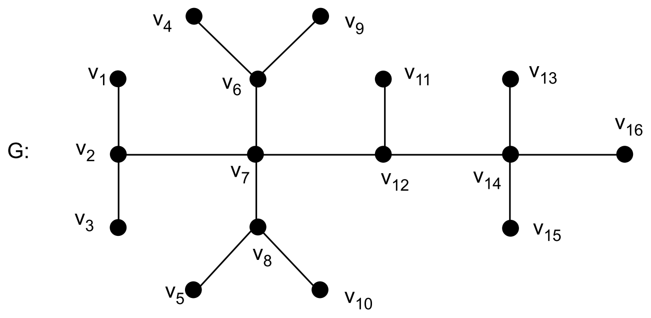
Let V be the vertex set of the graph G. The diameter of G is denoted as diam (G) and is defined as

$$\text{diam}(G) = \text{Max}\{e(v): v \in V\}.$$

## ■ 13.6 CENTRAL POINT AND CENTRE

Let V be the vertex set of the graph G. Then $v \in V$ is said to be a central point if $e(v) = \text{rad}(G)$. The set of all central points of G is known as centre of G.

Consider the following graph G where each edge is of length 1.



Here V= $\{v_1, v_2, v_3, \ldots, v_{16}\}$. Now, we get

$d(v_1, v_2) = 1$; $d(v_1, v_3) = 2$; $d(v_1, v_4) = 4$; $d(v_1, v_5) = 4$; $d(v_1, v_6) = 3$;
$d(v_1, v_7) = 2$; $d(v_1, v_8) = 3$; $d(v_1, v_9) = 4$; $d(v_1, v_{10}) = 4$; $d(v_1, v_{11}) = 4$;
$d(v_1, v_{12}) = 3$; $d(v_1, v_{13}) = 5$; $d(v_1, v_{14}) = 4$; $d(v_1, v_{15}) = 5$; $d(v_1, v_{16}) = 3$.

Therefore, $e(v_1) = \text{Max}\{1, 2, 3, 4, 5\} = 5$.

$d(v_2, v_1) = 1$; $d(v_2, v_3) = 1$; $d(v_2, v_4) = 3$; $d(v_2, v_5) = 3$; $d(v_2, v_6) = 2$; $d(v_2, v_7) = 1$;
$d(v_2, v_8) = 2$; $d(v_2, v_9) = 3$; $d(v_2, v_{10}) = 3$; $d(v_2, v_{11}) = 3$; $d(v_2, v_{12}) = 2$; $d(v_2, v_{13}) = 4$;
$d(v_2, v_{14}) = 3$; $d(v_2, v_{15}) = 4$; $d(v_2, v_{16}) = 4$.

Therefore, $e(v_2) = \text{Max}\{1, 2, 3, 4\} = 4$. Proceeding in this manner, we will get

$e(v_3) = 5$; $e(v_4) = 5$; $e(v_5) = 5$; $e(v_6) = 4$; $e(v_7) = 3$;
$e(v_8) = 4$; $e(v_9) = 5$; $e(v_{10}) = 5$; $e(v_{11}) = 4$; $e(v_{12}) = 3$;
$e(v_{13}) = 5$; $e(v_{14}) = 4$; $e(v_{15}) = 5$; $e(v_{16}) = 5$.

Now, radius = rad (G) = Min $\{e(v), v \in V\}$ = Min $\{5, 4, 3\}$
= 3 and
diameter = diam (G) = Max $\{e(v), v \in V\}$
= Max $\{5, 4, 3\} = 5$.

Therefore, the central points are $v_7$ and $v_{12}$ and center = $\{v_7, v_{12}\}$.

## ■ 13.7 SPANNING TREE

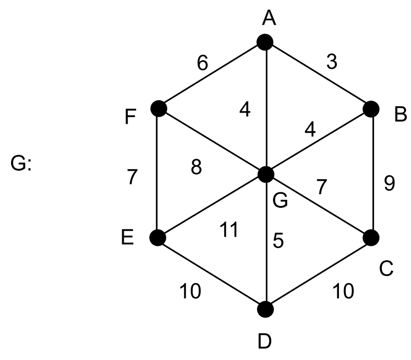
Suppose G = (V, E) be a graph. A sub graph H of G is said to be a spanning sub graph of G if both H and G has same vertex set. A spanning tree of a graph G is a tree which is a spanning sub graph of G. In this section we will discuss the algorithms for finding minimum spanning tree.

### 13.7.1 Prim's Algorithm

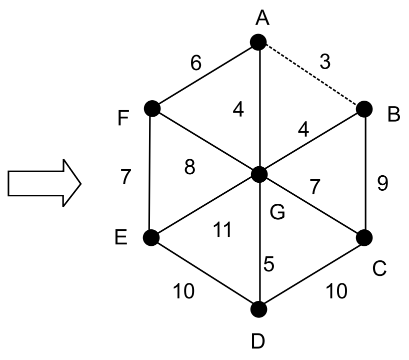
The following steps are used in Prim's algorithm for finding a minimum spanning tree of a graph G. Assume that the graph G has $n$ vertices.

1. Choose any vertex $v_1$ of G
2. Choose an edge $e_1 = v_1 v_2$ of G such that $v_1 \neq v_2$ and $e_1$ has smallest weight among the edges of G incident with $v_1$.
3. If edges $e_1, e_2, \ldots\ldots\ldots, e_i$ have been chosen involving vertices $v_1, v_2, \ldots, v_{i+1}$, then choose an edge $e_{i+1} = u\ v$ with $u \in \{v_1, v_2, \ldots, v_{i+1}\}$ and $v \notin \{v_1, v_2, \ldots, v_{i+1}\}$ such that $e_{i+1}$ has smallest weight among the edges of G.
4. The step 3 is to be repeated until we are getting the total $(n - 1)$ edges.

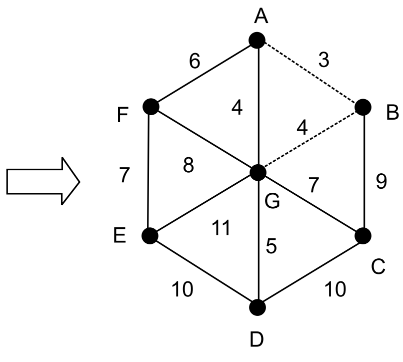
Consider the following connected weighted graph G. Here the number of vertices $n = 7$



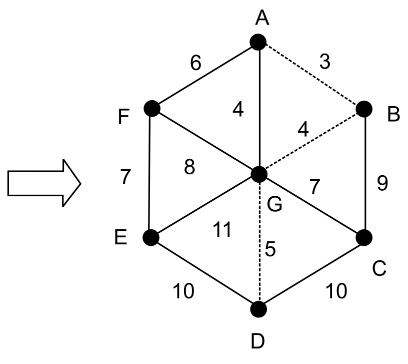
($V_1$= A ; One can choose any other vertex)



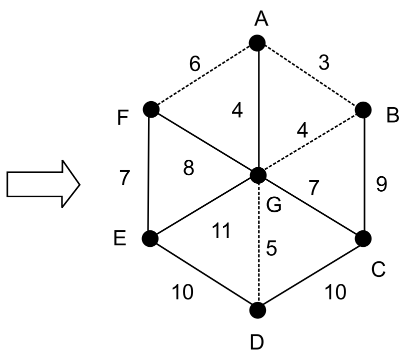
($e_1$ = AB, So that $v_2$ = B)



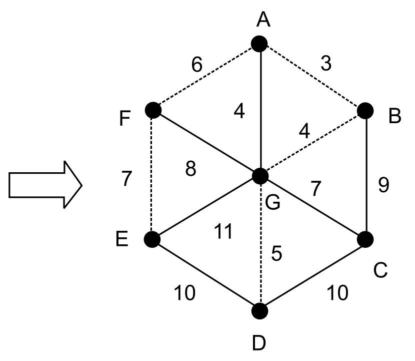
($e_2$ = BG ; So that $v_3$ = G ; An alternative choice is AG)



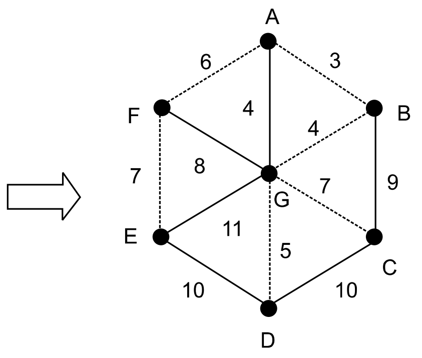
($e_3$ = GD ; So that $v_4$ = D; No alternative choice)



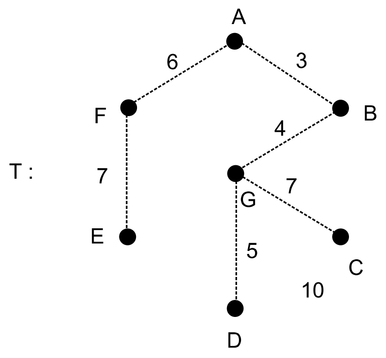
($e_4$ = AF ; So that $v_4$ = F ; No alternative choice)



($e_5$ = FE ; So that $v_5$ = E ; An alternative choice is GC)



($e_6$ = GC ; So that $v_6$ = C; No alternative choice)



$w$(T) = 32

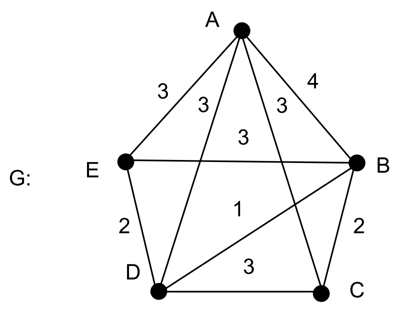
Since the total edges are 6 = (7 – 1), the process terminates. Hence, the minimum spanning tree T is given as shown in the above figure.

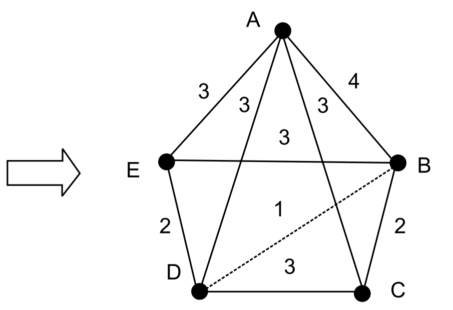
### 13.7.2 Kruskal's Algorithm

The following steps are used in Kruskal's algorithm for finding a minimum spanning tree of a graph G. Assume that the graph G has $n$ vertices.

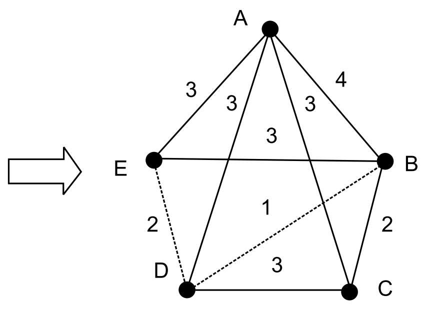
1. Choose an edge $e_1$ of G, which is as small as possible and $e_1$ must not be a loop.
2. Suppose the edges $e_1, e_2, ..., e_k$ have been chosen. Then the edge $e_{k+1}$ (not already chosen) can be chosen such that
   (*i*) The induced sub graph G[{$e_1, e_2$ ,.......... , $e_{k+1}$}] is acyclic and
   (*ii*) Weight of $e_{k+1}$ is as small as possible.
3. The step 2 is to be repeated until we are getting the total ($n$ – 1) edges.

Consider the following connected weighted graph G. Here the number of vertices $n$ = 5. On applying Kruskal's algorithm we have the following stages.

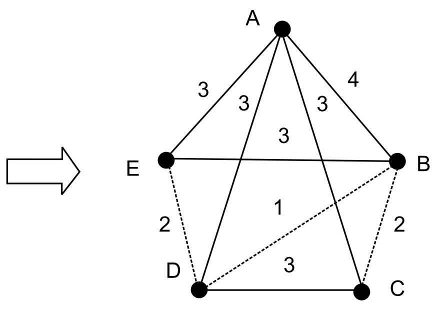




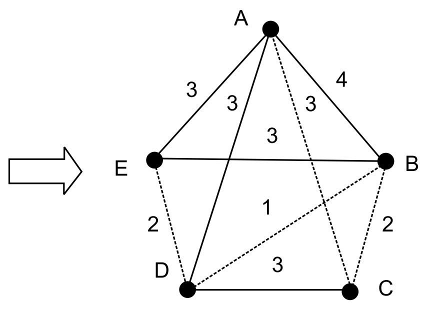
($e_1$ = BD ; $w$ ($e_1$) = 1 ; No alternative choice)



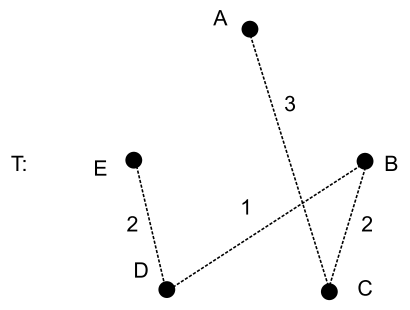
($e_2$ = DE ; $w$ ($e_2$) = 2; An alternative choice is BC)



($e_3$ = BC; $w$ ($e_3$) = 2 ; No alternative choice)



($e_4$ = AC; $w$ ($e_4$) = 3 ; Alternative choices are AE and AD)



$w$ (T) = 1 + 2 + 2 + 3 = 8

Since the total edges are 4 = (5 – 1), the process terminates. Hence, the minimum spanning tree T is given as shown in the above figure.

## ■ 13.8 SEARCHING ALGORITHMS

This section presents methods for searching a graph. This means systematically following the edges of the graph so as to visit the vertices of the graph. The graph searching algorithms can discover much about the structure of a graph. Here we present two algorithms, depth first search and breadth first search. In addition, we will discuss to create a breadth first and depth first tree.

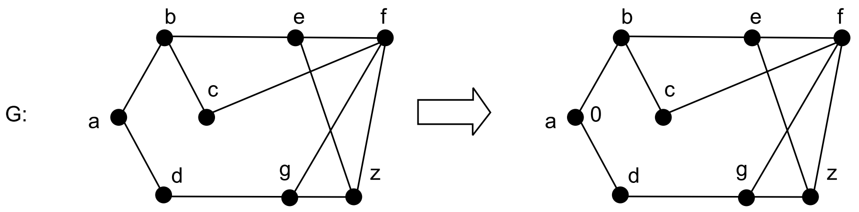
### 13.8.1 Breadth First Search

Breadth first search is one of the simplest algorithms for searching a graph. Given a graph G(V, E) and a distinguished source vertex *s*, breadth first search systematically explores the edges of G to discover every vertex that is reachable from *s*. It computes the distance ( fewest number of edges) from *s* to all such reachable vertices. Breadth first search is so named because it expands the frontier between discovered and undiscovered vertices uniformly across the breadth of the frontier. It constructs a breadth first tree, initially containing only its root, that is the source vertex *s*.

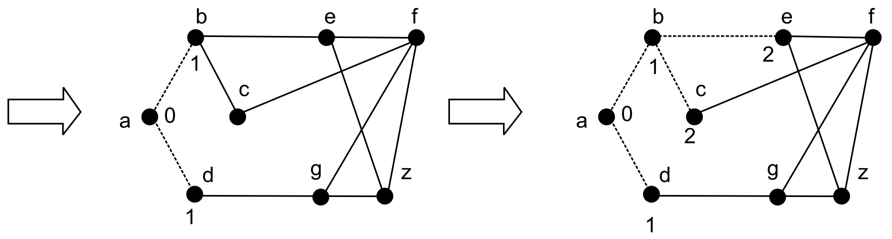
Suppose that $v_i$, $v_j$, be two specified vertices of G. We will now describe a method of finding a path from $v_i$ to $v_j$ which uses the least number of edges. Such a path is known as shortest path, if it exists. The method assigns labels 0, 1, 2, ... to the vertices of G and is called the Breadth First Search (BFS) technique. The BFS algorithm consists of the following steps:

1. Label the vertex $v_i$ with 0. Set $i = 0$
2. Find all unlabelled vertices in G, which are adjacent to vertices, labeled $i$. If there are no such vertices, then $v_i$ is not connected to $v_j$ else label them by $(i + 1)$.
3. If $v_j$ is labeled go to step 4, else replace $i$ by $(i + 1)$ and go to step 2.
4. The length of shortest path from $v_i$ to $v_j$ is $(i + 1)$ then stop.

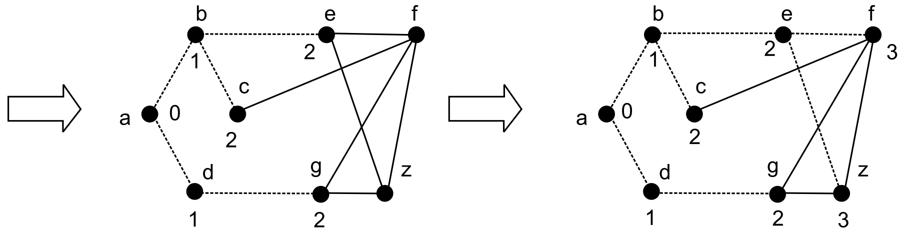
Consider the following graph G. Now we have to find out the shortest path from the source vertex $a$ to the vertex $z$. On using the BFS technique, we get the following stages.



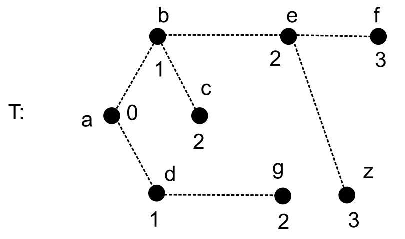
(Label $(a) = 0$ and set $i = 0$)



In the above figure the adjacent vertices of $a$ are $b$ and $d$. Therefore we get label $(b) = i + 1 = 0 + 1 = 1$ and label $(d) = i + 1 = 0 + 1 = 1$. Similarly, adjacent vertices of $b$ are $c$ and $e$. Therefore, label $(c) = i + 1 = 1 + 1 = 2$ and label $(e) = i + 1 = 1 + 1 = 2$.



In the above figure the adjacent vertices of $d$ is $g$. Therefore we get label $(g) = i + 1 = 1 + 1 = 2$. Similarly, adjacent vertices of $e$ are $f$ and $z$. Therefore, label $(f) = i + 1 = 2 + 1 = 3$ and label $(z) = i + 1 = 2 + 1 = 3$. Hence, the breadth first tree T becomes

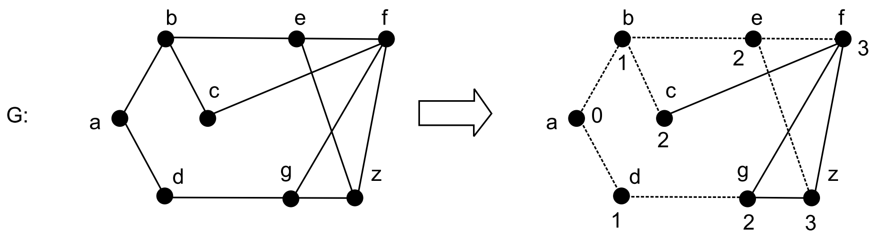


## 13.8.2 Back-Tracking Algorithm

The following steps are used in back-tracking algorithm:

1. Set $\lambda(t) = i$ and assign $v_i = t$, where '$t$' is the terminating node.
2. Find a vertex '$u$' which is adjacent to $v_i$ and with $\lambda(u) = (i - 1)$. Set $v_{i-1} = u$.
3. If $i = 1$, then stop else replace $i$ by $(i - 1)$ and go to step 2.

Consider the following graph G. Now we have to find out the shortest path from the source vertex '$a$' to the vertex '$z$'. On using the BFS technique, we get.



On using back-tracking algorithm we have

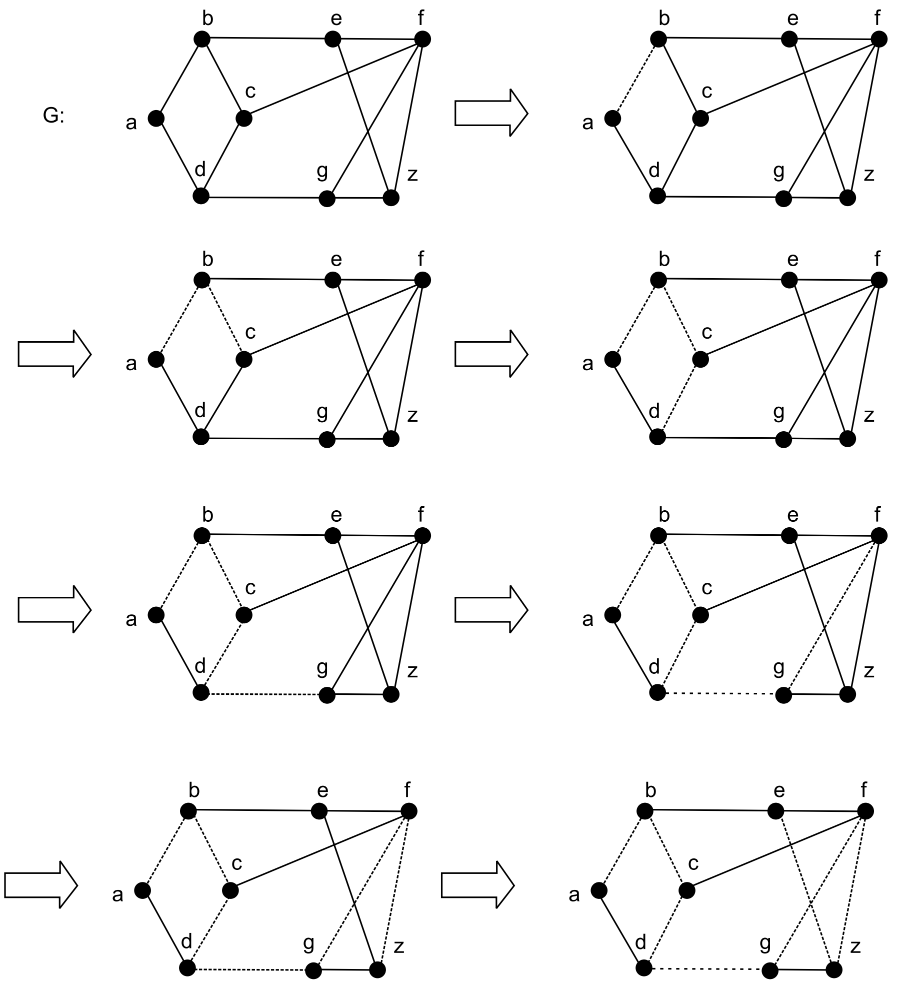
1. Set $i = \lambda(z) = 3$ and $v_i = v_3 = z$
2. The adjacent to $v_3 = z$ is $e$ and $\lambda(e) = (i - 1) = 2$. Set $v_2 = e$.
3. As $i = 3 \neq 1$, so $i = (i - 1) = 2$, Go to step 2.
   2. The adjacent to $v_2 = e$ is $b$ and $\lambda(b) = (i - 1) = 1$. Set $v_1 = b$.
   3. As $i = 2 \neq 1$, so $i = (i - 1) = 1$, Go to step 2.
      2. The adjacent to $v_1 = b$ is $a$ and $\lambda(a) = (i - 1) = 0$. Set $v_0 = a$.
      3. As $i = 1$, so the process terminates.

Therefore, the shortest path from '$a$' to $z$ is given as '$a\ b\ e\ z$'. Besides that, there could be several paths from '$a$' to '$z$'.

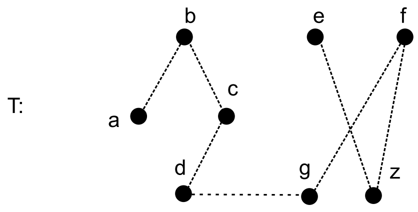
## 13.8.3 Depth First Search

Basic philosophy in depth first search is that first all vertices reachable from the vertex '$v$' are searched before proceeding to see the siblings. In depth first search, edges are explored out of the most recently discovered vertex '$v$' that still has unexplored edges leaving it. When all of $v$'s edges have been explored, then the search "backtracks" to explore edges leaving the vertex from which '$v$' was discovered. The process is being continued until we have discovered all the vertices that are reachable from the original source vertex. If any undiscovered vertices remain, then one of them is selected as a new source vertex and the search is repeated. This process is repeated until all vertices are discovered.

Consider the graph G as below. Let us consider the source vertex as '*a*'. On using the DFS technique, the order in which the vertices are being visited is described below by the sequence of graphs.



Therefore, the depth first tree T is given below. Besides that, there could be several depth first trees from the same vertex '*a*'. This indicates that the depth first tree is not unique.



## ■ 13.9 SHORTEST PATH ALGORITHMS

This section presents methods for finding shortest path from a source vertex to a terminating vertex in a graph G. This problem is a real life problem, where cities are connected through roads, rails and air routes and we want to find out the shortest path between the vertices. Here we present two algorithms Dijkastra's Algorithm and Floyd-Warshall Algorithm.

### 13.9.1 Dijkstra's Algorithm

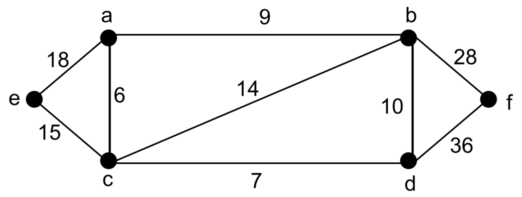
In a given graph G(V, E), we want to find a shortest path from a given source vertex '$s \in V$' to every vertex '$v \in V$'. This is otherwise known as single source shortest path problem. Dijkastra's Algorithm solves the single source shortest path problems in weighted graphs that to non-negative weights. Therefore, we assume that $w(uv) \geq 0$ for each edge $(uv) \in E$.

1. Set $\lambda(v_s) = 0$ and for all vertices $v_i \neq v_s$ $\lambda(v_i) = \infty$. Set T = V; where V is the set of vertices of G and T is the set of uncoloured vertices.
2. Let $u$ be the vertex in T for which $\lambda(u)$ is minimum.
3. If $u = v_t$ (Terminating node), then stop. Else, go to step 4.
4. For every edge $e = uv$, incident with $u$, if $v \in T$, then replace $\lambda(v)$ with Min $\{\lambda(v), \lambda(u) + w(uv)\}$.

*i.e,* $\lambda(v) = \text{Min}\,\{\lambda(v), \lambda(u) + w(uv)\}$

5. Change T by T – $\{u\}$ and go to step 2.

Consider the following graph G. Let us consider the source vertex as *e* and the terminating vertex as *f*. We have to find out the shortest distance between the vertices *e* and *f*.



In the above graph G, the source vertex is $v_s = e$ and $v_t = f$. Set $\lambda(e) = 0$ and $\lambda(a) = \lambda(b) = \lambda(c) = \lambda(d) = \lambda(f) = \infty$. T = V = $\{e, a, b, c, d, f\}$. Hence, we have the following table

| Vertex | *e* | *a* | *b* | *c* | *d* | *f* |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T | *e* | *a* | *b* | *c* | *d* | *f* |

Now, $u = e$ as $\lambda(u) = \lambda(e) = 0$ which is minimum. The edges incident on $u = e$ are *ea* and *ec*. Therefore,

$$\begin{aligned}\lambda(a) &= \text{Min}\,[\lambda(a), \lambda(e) + w(ea)] \\ &= \text{Min}\,[\infty, 18] = 18. \\ \lambda(c) &= \text{Min}\,[\lambda(c), \lambda(e) + w(ec)] \\ &= \text{Min}\,[\infty, 15] = 15.\end{aligned}$$

Again, $T = T - \{u = e\} = \{a, b, c, d, f\}$. Thus, we have the following table

| Vertex | *e* | *a* | *b* | *c* | *d* | *f* |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 18 | ∞ | 15 | ∞ | ∞ |
| T | | *a* | *b* | *c* | *d* | *f* |

Now, $u = c$ as $\lambda(u) = \lambda(c) = 15$ which is minimum. The edges incident with $u = c$ are $ca$, $cb$ and $cd$. Therefore,

$$\lambda(a) = \text{Min}\ [\lambda(a), \lambda(c) + w(ca)]$$
$$= \text{Min}\ [18, 21] = 18.$$
$$\lambda(b) = \text{Min}\ [\lambda(b), \lambda(c) + w(cb)]$$
$$= \text{Min}\ [\infty, 29] = 29.$$
$$\lambda(d) = \text{Min}\ [\lambda(d), \lambda(c) + w(cd)]$$
$$= \text{Min}\ [\infty, 22] = 22.$$

Again, $T = T - \{u = c\} = \{a, b, d, f\}$. Thus, we have the following table

| Vertex | $e$ | $a$ | $b$ | $c$ | $d$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 18 | 29 | 15 | 22 | $\infty$ |
| T | | $a$ | $b$ | | $d$ | $f$ |

Now, $u = a$ as $\lambda(u) = \lambda(a) = 18$ which is minimum. The edges incident with $u = a$ is $ab$. Therefore,

$$\lambda(b) = \text{Min}\ [\lambda(b), \lambda(a) + w(ab)]$$
$$= \text{Min}\ [29, 27] = 27.$$

Again, $T = T - \{u = a\} = \{b, d, f\}$. Thus, we have the following table

| Vertex | $e$ | $a$ | $b$ | $c$ | $d$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 18 | 27 | 15 | 22 | $\infty$ |
| T | | | $b$ | | $d$ | $f$ |

Now, $u = d$ as $\lambda(u) = \lambda(d) = 22$ which is minimum. The edges incident with $u = d$ are $db$ and $df$. Therefore,

$$\lambda(b) = \text{Min}\ [\lambda(b), \lambda(d) + w(db)]$$
$$= \text{Min}\ [27, 32] = 27.$$
$$\lambda(f) = \text{Min}\ [\lambda(f), \lambda(d) + w(df)]$$
$$= \text{Min}\ [\infty, 58] = 58.$$

Again, $T = T - \{u = d\} = \{b, f\}$. Thus, we have the following table.

| Vertex | $e$ | $a$ | $b$ | $c$ | $d$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 18 | 27 | 15 | 22 | 58 |
| T | | | $b$ | | | $f$ |

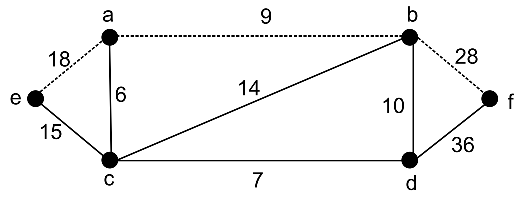
Now, $u = b$ as $\lambda(u) = \lambda(b) = 27$ which is minimum. The edges incident with $u = b$ is $bf$. Therefore,

$$\lambda(f) = \text{Min}\ [\lambda(f), \lambda(b) + w(bf)]$$
$$= \text{Min}\ [58, 55\ ] = 55.$$

Again, $T = T - \{u = b\} = \{f\}$. Thus, we have the following table

| Vertex | $e$ | $a$ | $b$ | $c$ | $d$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 18 | 27 | 15 | 22 | 55 |
| T | | | | | | $f$ |

Now, $u = f$ and $f$ is the terminating node, so the process terminates. Hence, the shortest distances from $e$ to $a$, $b$, $c$, $d$ and $f$ are 18, 27, 15, 22, 55 respectively. The shortest distance between $e$ and $f$ is given in the following figure.



### 13.9.2 Floyd-Warshall Algorithm

Floyd-Warshall algorithm solves all-pairs shortest paths problem on a directed weighted graph G = (V, E). The weighted graph may contain negative weight edges, but we shall assume that there are no negative weight cycles. In this algorithm, we use the adjacency matrix of the graph to find out the shortest distance between any pair of vertices.

Suppose that G(V, E) be a graph. Let W be the adjacency matrix of the weighted directed graph G. The algorithm has the following steps:

```
1. $n$ = Rows [W]
2. $D^{(0)} = W$
3. For $k = 1$ to $n$
4. Do for $i = 1$ to $n$
5. Do for $j = 1$ to $n$
6. $d_{ij}^{(k)} = \text{Min}\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
7. Write $D^{(n)}$
```

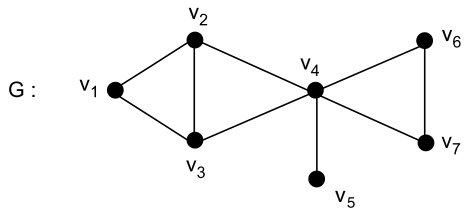
where, $D^{(n)} = \left(d_{ij}^{n}\right)$.

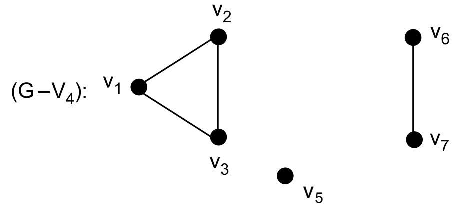
## 13.10 CUT VERTICES

Suppose that G(V, E) be the graph. A vertex '$v$' of a graph G is called a cut vertex of G if the number of component of $(G - v)$ is greater than the number of components of G.

*i.e.* $w\,(G - v) > w(g)$, where $w(G)$ represents number of component of G.

Consider the graph G as below. In the graph G, '$v_4$' is a cut vertex as $w(G - v_4) = 3 > w(G) = 1$.





## ■ 13.11 EULER GRAPH

A tour is a closed walk of G, which include every edge of G at least once. An Euler tour is a closed walk of G, which include every edge of G exactly once. A graph G is said to be an Euler or Eulerian if the graph G has an Euler tour.

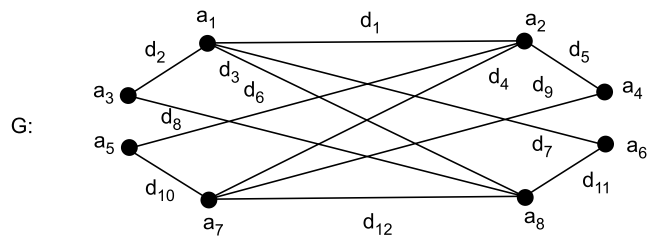
In this section, we will discuss two algorithms *i.e.,* Fleury's algorithm and Hierholzer's algorithm to construct Euler tour in a Euler graph.

### 13.11.1 Fleury's Algorithm

This algorithm is generally developed to construct an Euler tour in a Euler graph. The following steps are used in this algorithm:

1. Choose any vertex $v_0$ in the Euler graph G and set $W_0 = v_0$.
2. If the trail $W_i = v_0\, e_1\, v_1\, e_2\, v_2 \ldots e_i\, v_i$ has been chosen, then choose an edge $e_{i+1}$ different from $e_1, e_2, \ldots, e_i$ such that
   (*i*) $e_{i+1}$ is incident with $v_i$ and
   (*ii*) unless there is no alternative, $e_{i+1}$ is not a bridge of the edge deleted subgraph $G - \{e_1, e_2, \ldots, e_i\}$
3. Stop if $w_i$ contains every edge of G; otherwise go to step 2.

Consider the following Euler graph G. We have to find out the Euler tour using Fleury's algorithm for the Euler graph G.



1. Let us choose $v_0 = a_1$ and set $w_0 = a_1$
2. Choose edge $e_1 = d_1$ such that $W_1 = v_0\, e_1 = a_1\, d_1\, a_2$
3. As $W_1$ contains only one edge, so go to step 2.
   2. Choose edge $e_2 = d_6$ such that $W_2 = a_1\, d_1\, a_2\, d_6\, a_5$
   3. As $W_2$ contains 2 edges, so go to step 2.
      2. Choose edge $e_3 = d_{10}$ such that $W_3 = a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7$
      3. As $W_3$ contains 3 edges, so go to step 2.
         2. Choose edge $e_4 = d_9$ such that $W_4 = a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_9\, a_4$
         3. As $W_4$ contains 4 edges, so go to step 2.
            2. Choose edge $e_5 = d_5$ such that $W_5 = a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_9\, a_4\, d_5\, a_2$
            3. As $W_5$ contains 5 edges, so go to step 2.

Proceeding in this manner, we will get

$$W_{12} = a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_9\, a_4\, d_5\, a_2\, d_4\, a_7\, d_{12}\, a_8\, d_8\, a_3\, d_2\, a_1\, d_3\, a_8\, d_{11}\, a_6\, d_7\, a_1$$

As $W_{12}$ contains all the 12 edges once, so the process terminates. Thus the Euler tour produced is given as

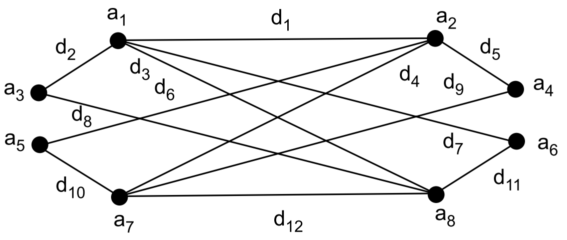
$$a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_9\, a_4\, d_5\, a_2\, d_4\, a_7\, d_{12}\, a_8\, d_8\, a_3\, d_2\, a_1\, d_3\, a_8\, d_{11}\, a_6\, d_7\, a_1$$

### 13.11.2 Hierholzer's Algorithm

Like Fleury's algorithm, this algorithm is also developed to construct an Euler tour in a Euler graph. The following steps are used in this algorithm:

1. Choose any vertex $v$ in G and choose any closed trail $W_0$ in G. Set $i = 0$.
2. If $E(W_i) = E(G)$, then stop and $W_i$ is an Euler tour of G; else chose a vertex $v_i$ on $W_i$ which is incident with an edge in G but not in $W_i$. Choose a closed trail $W_i$*in the subgraph G - $E(W_i)$, starting at the vertex $v_i$. Where $W_i^*$ is the detour trail.
3. Let $W_{i+1}$ be the closed trail consisting of the edges of both $W_i$ and $W_i^*$ obtained by starting at the vertex $v$, traversing the trail $W_i$ until $v_i$ is reached, then traversing the closed trail $W_i^*$ and returning to $v_i$ , completing the rest of the trail $W_i$. Replace $i$ by $(i + 1)$ and go to step 2.

Consider the graph G. We have to find out the Euler tour using Hierholzer's algorithm for the Euler graph G.

G:


1. Let $v = a_1$ choose the closed trail $W_0$ as $W_0 = a_1\, d_1\, a_2\, d_5\, a_4\, d_9\, a_7\, d_{12}\, a_8\, d_3\, a_1$. Set $i = 0$.
2. As $E(W_0) \neq E(G)$, choose $a_2$ on $W_0$ incident with $d_6$ not in $W_0$. Choose $W_0^* = a_2\, d_6\, a_5\, d_{10}\, a_7\, d_4\, a_2$; where all $d_i \in G - E(W_0)$; $i = 6, 4, 10$.
3. Now, we have $W_1 = W_{0+1} = a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_4\, a_2\, d_5\, a_4\, d_9\, a_7\, d_{12}\, a_8\, d_3\, a_1$ and $i = (i + 1) = 0 + 1 = 1$. Go to step 2.
   2. As $E(W_1) \neq E(G)$, choose $a_1$ on $W_1$ incident with $d_2$ not in $W_1$. Choose $W_1^* = a_1\, d_2\, a_3\, d_8\, a_8\, d_{11}\, a_6\, d_7\, a_1$; where all $d_i \in G - E(W_1)$; $i = 2, 8, 11, 7$.
   3. Now, we get $W_2 = W_{1+1} = a_1\, d_2\, a_3\, d_8\, a_8\, d_{11}\, a_6\, d_7\, a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_4\, a_2\, d_5\, a_4\, d_9\, a_7\, d_{12}\, a_8\, d_3 a_1$ and $i = (i + 1) = 2$. Go to step 2. Since, $E(W_2) = E(G)$; the process terminates. Therefore, the Euler tour is given as

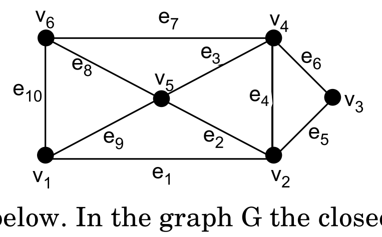
$$a_1\, d_2\, a_3\, d_8\, a_8\, d_{11}\, a_6\, d_7\, a_1\, d_1\, a_2\, d_6\, a_5\, d_{10}\, a_7\, d_4\, a_2\, d_5\, a_4\, d_9\, a_7\, d_{12}\, a_8\, d_3\, a_1.$$

### 13.11.3 Euler Trail

Suppose that G be the graph. A trail in G is said to be an Euler trail if it contains every edge of G exactly once. So every Euler tour is a closed Euler Trail.

Consider the graph G as below. One Euler trail in the graph G is given as

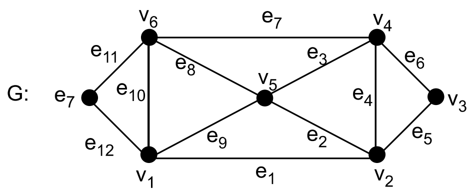
$$v_1\, e_1\, v_2\, e_2\, v_5\, e_3\, v_4\, e_4\, v_2\, e_5\, v_3\, e_6\, v_4\, e_7\, v_6\, e_8\, v_5\, e_9\, v_1\, e_{10}\, v_6.$$

G:


Consider another graph G as below. In the graph G the closed Euler trail is given as

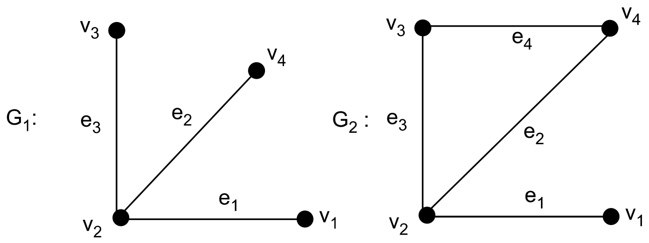
$$v_1\, e_1\, v_2\, e_2\, v_5\, e_3\, v_4\, e_6 v_3\, e_5\, v_2\, e_4\, v_4\, e_7\, v_6\, e_8\, v_5\, e_9\, v_1\, e_{10}\, v_6\, e_{11}\, v_7\, e_{12}\, v_1.$$

This is known as an Euler Tour.



## ■ 13.12 HAMILTONIAN PATH

A path of a graph G(V, E) which contains every vertex of G exactly once is known as Hamiltonian path. Consider the following graphs:

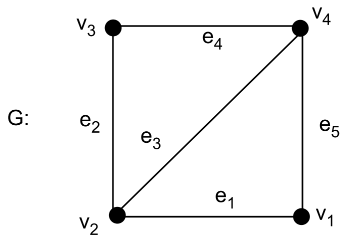


The graph $G_1$ has no Hamiltonian path where as $G_2$ has a Hamiltonian path, *i.e.*,

$$v_1\, e_1\, v_2\, e_3\, v_3\, e_4\, v_4 \,.$$

### 13.12.1 Hamiltonian Graph

A cycle in a graph G, which contains every vertex of G only once, is known as a Hamiltonian cycle. It is to be noted that no vertex of a cycle is repeated apart from the final vertex, which is same as the starting vertex. A graph G is said to be Hamiltonian if it has a Hamiltonian cycle. Consider the graph G as

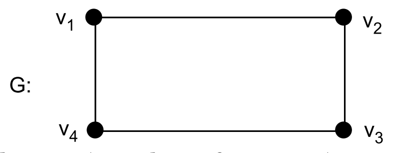


The Hamiltonian cycle is $v_1\, e_1\, v_2\, e_2\, v_3\, e_4\, v_4\, e_5\, v_1$. Therefore, the graph G is a Hamiltonian graph.

## ■ 13.13 CLOSURE OF A GRAPH

Let G be a simple graph. If there are two non-adjacent vertices $u_1$ and $v_1$ in G such that $d(u_1) + d(v_1) \geq n$ (number of vertices in G) then join $u_1$ and $v_1$ by an edge to get the super graph $G_1$ of G. Continue this process recursively joining pairs of non-adjacent vertices whose degree sum is at least $n$ until no such pair remains. The final super graph thus obtained is called the closure of G denoted by C(G).

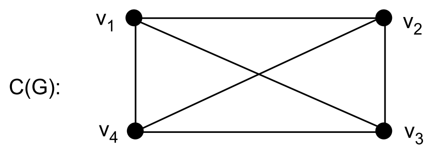
Consider the graph G as



Here, V = { $v_1$, $v_2$, $v_3$, $v_4$ } and $n = 4$ (number of vertices). Now for the non-adjacent vertices $v_1$ and $v_3$ we get

$$d(v_1) + d(v_3) = 2 + 2 = 4 \geq n = 4.$$

Therefore, there exists an edge between $v_1$ and $v_3$. Similarly, for the non-adjacent vertices $v_2$ and $v_4$ we get $d(v_2) + d(v_4) = 2 + 2 = 4 \geq 4 = n$. So, there exists an edge between $v_2$ and $v_4$. Thus, the final super graph is given as below. This is nothing but the closure of G, *i.e.*, C(G).



## ■ 13.14 TRAVELLING SALESMAN PROBLEM

The job of a travelling salesman is to visit all the towns linked with roads in a particular territory. He has to visit all the towns exactly once in such a manner that the total distance travelled by himself will be minimum.

In graph theory we denote nodes as towns joined by a weighted edge if and only if road connects them which does not pass through any of the other towns. In travelling salesman problem, we have to construct a minimum Hamiltonian cycle. The following algorithms provide minimum Hamiltonian cycle in case of a complete weighted graph:

(*i*) Two optimal algorithm and
(*ii*) Closest insertion algorithm

### 13.14.1 Two-Optimal Algorithm

Suppose that G(V, E) be a complete weighted graph. Where V{$v_1$, $v_2$, ... , $v_n$}. Here we choose a Hamiltonian cycle C and perform a sequence of modifications to C to find a smaller weight. The following steps are used in two-optimal algorithm:

1. Let C = $v_1v_2$ ...... $v_nv_1$ be a Hamiltonian cycle of the complete weighted graph G. Calculate the weight $w$ of C by the relation
$$w = w(v_1, v_2) + w(v_2, v_3) + ... + w(v_n, v_1).$$
Where, $w(v_i, v_j)$ denote the weight of the edge joining $v_i$ and $v_j$.
2. Set $i = 1$
3. Set $j = i + 2$
4. Let $C_{ij}$ denote the Hamiltonian cycle as
$$C_{ij} = v_1v_2v_3 \ldots v_iv_jv_{j-1}\,v_{j-2} \ldots v_{i+1}\,v_{j+1} \ldots v_nv_1.$$

Calculate $w_{ij}$ of $C_{ij}$, where $w_{ij} = w - w(v_i\, v_{i+1}) - w(v_j\, v_{j+1}) + w(v_i\, v_j) + w(v_{i+1}\, v_{j+1})$.

5. If $w_{ij} < w$, then replace C by $C_{ij}$ and $w$ by $w_{ij}$ . Also relabel the vertices of $C_{ij}$ in the order $v_1v_2v_3$ ..... $v_nv_1$; else go to step 6.
6. Set $j = (j + 1)$. If $j \leq n$, go to step 4 else set $i = (i + 1)$.
7. If $i \leq (n - 2)$, go to step 3 else stop.

### 13.14.2 The Closest Insertion Algorithm

In this algorithm we gradually build up a sequence of cycles in the graph which involve more and more vertices until all the vertices are chosen up. In this case one more vertex is inserted into the cycle each time in cheapest possible way. The description uses the idea of the distance of a vertex $v$ from a walk W. The following steps are used in this algorithm:

1. Choose any vertex $v_1$ as a starting vertex.
2. Choose the 2nd vertex $v_2$ which is closest to $v_1$ from the $(n - 1)$ vertices not chosen so far. Let $w_2 = v_1v_2v_1$ denote the walk.
3. Choose the 3rd vertex $v_3$ which is closest to the walk $w_2 = v_1v_2v_1$ from the $(n - 2)$ vertices not chosen so far. Let $w_3 = v_1v_2v_3v_1$ denote the walk.
4. Choose the 4th vertex $v_4$ which is closest to the walk $w_3 = v_1v_2v_3v_1$ from the $(n - 3)$ vertices not chosen so far. Find the shortest walk from the walks $v_1\, v_2\, v_3\, v_4\, v_1$; $v_1\, v_2\, v_4\, v_3\, v_1$; $v_1\, v_4\, v_2\, v_3\, v_1$. Let $w_4$ denote the shortest walk. Relabel the vertices as $v_1\, v_2\, v_3\, v_4\, v_1$ if necessary.
5. Choose the 5th vertex $v_5$ which is closest to the walk $w_4$ from the $(n - 4)$ vertices not chosen so far. Find the shortest walk from the walks $v_1\, v_2\, v_3\, v_4\, v_5\, v_1$; $v_1\, v_2\, v_3\, v_5\, v_4\, v_1$; $v_1\, v_2\, v_5\, v_3\, v_4\, v_1$; $v_1\, v_5\, v_2\, v_3\, v_4\, v_1$. Let $w_5$ denote the shortest walk. Relabel the vertices as $v_1\, v_2\, v_3\, v_4\, v_5\, v_1$ if necessary.
6. The process is being repeated until all the vertices are included in the cycle. Therefore the walk $w_n$ is the Hamiltonian cycle of the graph G.

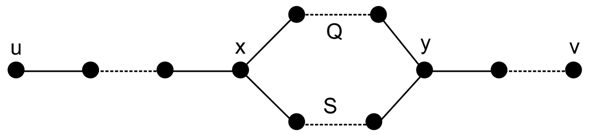
**Note:** Both the algorithms *i.e.,* Two optimal algorithm and Closest insertion algorithm provide reasonably good solutions. Therefore, both are approximately optimal.

## SOLVED EXAMPLES

**Example 1** *If u and v are distinct vertices of a tree T, then T contains exactly one u – v path.*

**Solution:** Suppose, to the contrary, the tree T contains two $u - v$ paths. Let us assume that the two $u - v$ paths are Q and S.

Since Q and S are different $u - v$ paths, there must exists a vertex $x$ belonging to both Q and S such that the vertex immediately following $x$ on Q is different from the vertex immediately following $x$ on S. This can be easily understandable from the figure shown below.

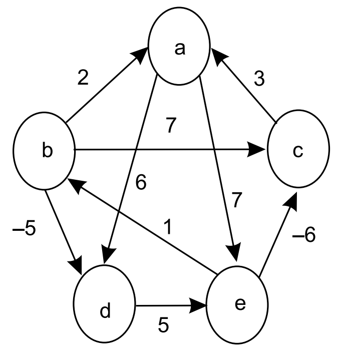


Let us assume that $y$ be the first vertex of Q following $x$, which also belongs to S. This implies that there exists two $x - y$ paths that have only $x$ and $y$ in common. It is clear that these two paths produce a cycle in T. This is a contradiction. This contradict to the fact that T is a tree.

Therefore, our supposition is wrong. Hence, T has only one $u - v$ path.

**Example 2** *For the following weighted graph G apply Floyd-Warshall algorithm to find the shortest path between any pair of vertices a, b, c, d and e. Show at least one iteration in details.*

G:



**Solution:** The adjacency matrix W with respect to the nodes $b$, $a$, $c$, $e$

and $d$ is given as $\begin{pmatrix} 0 & 2 & 7 & \infty & -5 \\ \infty & 0 & \infty & 7 & 6 \\ \infty & 3 & 0 & \infty & \infty \\ 1 & \infty & -6 & 0 & \infty \\ \infty & \infty & \infty & 5 & 0 \end{pmatrix}$

Hence, $n = \text{Row}[W] = 5$

$$D^{(0)} = \left(d_{ij}^{(0)}\right) = \begin{pmatrix} 0 & 2 & 7 & \infty & -5 \\ \infty & 0 & \infty & 7 & 6 \\ \infty & 3 & 0 & \infty & \infty \\ 1 & \infty & -6 & 0 & \infty \\ \infty & \infty & \infty & 5 & 0 \end{pmatrix}$$

For $k = 1$, $i = 1$ and $j = 1$ to 5 we get

$$d_{11}^1 = \text{Min}(d_{11}^0, d_{11}^0 + d_{11}^0) = \text{Min}(0, 0+0) = 0$$
$$d_{12}^1 = \text{Min}(d_{12}^0, d_{11}^0 + d_{12}^0) = \text{Min}(2, 0+2) = 2$$
$$d_{13}^1 = \text{Min}(d_{13}^0, d_{11}^0 + d_{13}^0) = \text{Min}(7, 0+7) = 7$$
$$d_{14}^1 = \text{Min}(d_{14}^0, d_{11}^0 + d_{14}^0) = \text{Min}(\infty, 0+\infty) = \infty$$
$$d_{15}^1 = \text{Min}(d_{15}^0, d_{11}^0 + d_{15}^0) = \text{Min}(-5, 0-5) = -5$$

For $k = 1$, $i = 2$ and $j = 1$ to 5 we get

$$d_{21}^1 = \text{Min}(d_{21}^0, d_{21}^0 + d_{11}^0) = \text{Min}(\infty, \infty+0) = \infty$$
$$d_{22}^1 = \text{Min}(d_{22}^0, d_{21}^0 + d_{12}^0) = \text{Min}(0, \infty+2) = 0$$
$$d_{23}^1 = \text{Min}(d_{23}^0, d_{21}^0 + d_{13}^0) = \text{Min}(\infty, \infty+7) = \infty$$
$$d_{24}^1 = \text{Min}(d_{24}^0, d_{21}^0 + d_{14}^0) = \text{Min}(7, \infty+\infty) = 7$$
$$d_{25}^1 = \text{Min}(d_{25}^0, d_{21}^0 + d_{15}^0) = \text{Min}(6, \infty-5) = 6$$

For $k = 1$, $i = 3$ and $j = 1$ to $5$ we get

$$d_{31}^{1} = \text{Min}(d_{31}^{0}, d_{31}^{0} + d_{11}^{0}) = \text{Min}(\infty, \infty + 0) = \infty$$
$$d_{32}^{1} = \text{Min}(d_{32}^{0}, d_{31}^{0} + d_{12}^{0}) = \text{Min}(3, \infty + 2) = 3$$
$$d_{33}^{1} = \text{Min}(d_{33}^{0}, d_{31}^{0} + d_{13}^{0}) = \text{Min}(0, \infty + 7) = 0$$
$$d_{34}^{1} = \text{Min}(d_{34}^{0}, d_{31}^{0} + d_{14}^{0}) = \text{Min}(\infty, \infty + \infty) = \infty$$
$$d_{35}^{1} = \text{Min}(d_{35}^{0}, d_{31}^{0} + d_{15}^{0}) = \text{Min}(\infty, \infty - 5) = \infty$$

For $k = 1$, $i = 4$ and $j = 1$ to $5$ we get

$$d_{41}^{1} = \text{Min}(d_{41}^{0}, d_{41}^{0} + d_{11}^{0}) = \text{Min}(1, 1 + 0) = 1$$
$$d_{42}^{1} = \text{Min}(d_{42}^{0}, d_{41}^{0} + d_{12}^{0}) = \text{Min}(\infty, 1 + 2) = 3$$
$$d_{43}^{1} = \text{Min}(d_{43}^{0}, d_{41}^{0} + d_{13}^{0}) = \text{Min}(-6, 1 + 7) = -6$$
$$d_{44}^{1} = \text{Min}(d_{44}^{0}, d_{41}^{0} + d_{14}^{0}) = \text{Min}(0, 1 + \infty) = 0$$
$$d_{45}^{1} = \text{Min}(d_{45}^{0}, d_{41}^{0} + d_{15}^{0}) = \text{Min}(\infty, 1 - 5) = -4$$

For $k = 1$, $i = 5$ and $j = 1$ to $5$ we get

$$d_{51}^{1} = \text{Min}(d_{51}^{0}, d_{51}^{0} + d_{11}^{0}) = \text{Min}(\infty, \infty + 0) = \infty$$
$$d_{52}^{1} = \text{Min}(d_{52}^{0}, d_{51}^{0} + d_{12}^{0}) = \text{Min}(\infty, \infty + 2) = \infty$$
$$d_{53}^{1} = \text{Min}(d_{53}^{0}, d_{51}^{0} + d_{13}^{0}) = \text{Min}(\infty, \infty + 7) = \infty$$
$$d_{54}^{1} = \text{Min}(d_{54}^{0}, d_{51}^{0} + d_{14}^{0}) = \text{Min}(5, \infty + \infty) = 5$$
$$d_{55}^{1} = \text{Min}(d_{55}^{0}, d_{51}^{0} + d_{15}^{0}) = \text{Min}(0, \infty - 5) = 0$$

Therefore, we have
$$D^{(1)} = \begin{pmatrix} 0 & 2 & 7 & \infty & -5 \\ \infty & 0 & \infty & 7 & 6 \\ \infty & 3 & 0 & \infty & \infty \\ 1 & 3 & -6 & 0 & -4 \\ \infty & \infty & \infty & 5 & 0 \end{pmatrix}$$

Similarly for $k = 2$, $i = 1$ to $5$ and $j = 1$ to $5$, we get

$$D^{(2)} = \begin{pmatrix} 0 & 2 & 7 & 9 & -5 \\ \infty & 0 & \infty & 7 & 6 \\ \infty & 3 & 0 & 10 & 9 \\ 1 & 3 & -6 & 0 & -4 \\ \infty & \infty & \infty & 5 & 0 \end{pmatrix}$$

Similarly for $k = 3$, $i = 1$ to $5$ and $j = 1$ to $5$, we get

$$D^{(3)} = \begin{pmatrix} 0 & 2 & 7 & 9 & -5 \\ \infty & 0 & \infty & 7 & 6 \\ \infty & 3 & 0 & 10 & 9 \\ 1 & -3 & -6 & 0 & -4 \\ \infty & \infty & \infty & 5 & 0 \end{pmatrix}$$

Similarly for $k = 4$, $i = 1$ to 5 and $j = 1$ to 5, we get

$$D^{(4)} = \begin{pmatrix} 0 & 2 & 3 & 9 & -5 \\ 8 & 0 & 1 & 7 & 3 \\ 11 & 3 & 0 & 10 & 6 \\ 1 & -3 & -6 & 0 & -4 \\ 6 & 2 & -1 & 5 & 0 \end{pmatrix}$$

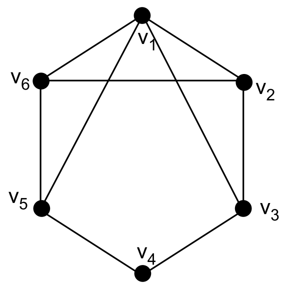
Similarly for $k = 5$, $i = 1$ to 5 and $j = 1$ to 5, we get

$$D^{(5)} = \begin{pmatrix} 0 & -3 & -6 & 0 & -5 \\ 8 & 0 & 1 & 7 & 3 \\ 11 & 3 & 0 & 10 & 6 \\ 1 & -3 & -6 & 0 & -4 \\ 6 & 2 & -1 & 5 & 0 \end{pmatrix}$$

From the above matrix, the shortest distance for any pair of vertices can be found out.

**Example 3** *Find the closure of the graph G where*

G:

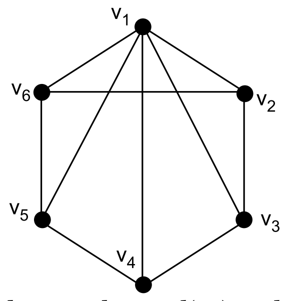


**Solution:** In the above graph G we have V = {$v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$} and number of vertices ($n$) = 6. Now for the non adjacent vertices $v_1$ and $v_4$ we have

$$d(v_1) + d(v_4) = 4 + 2 = 6 \geq n = 6.$$

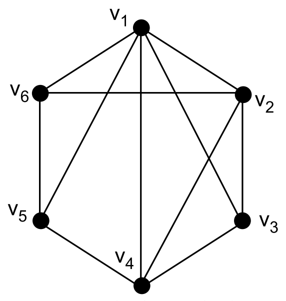
Therefore, there exists an edge between $v_1$ and $v_4$. Let the super graph $G_1$ be

$G_1$:



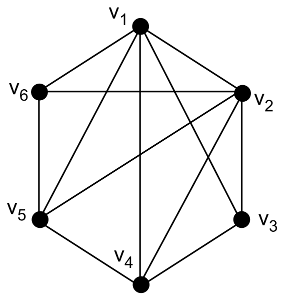
For the non-adjacent vertices $v_2$ and $v_4$ we have $d(v_2) + d(v_4) = 3 + 3 = 6 \geq n$ . Therefore, there exists an edge between $v_2$ and $v_4$ . Let the super graph $G_2$ be

G$_2$:



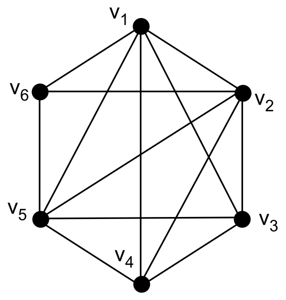
Again, $v_2$ and $v_5$ are non-adjacent such that $d(v_2) + d(v_5) = 4 + 3 = 7 \geq n = 6$. Therefore, there exists an edge between $v_2$ and $v_5$. Let the super graph $G_3$ be

G$_3$:



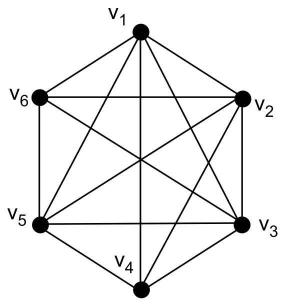
For the non-adjacent vertices $v_3$ and $v_5$ we have $d(v_3) + d(v_5) = 3 + 4 = 7 \geq n$. Therefore, there exists an edge between $v_3$ and $v_5$. Let the super graph $G_4$ be

G$_4$:

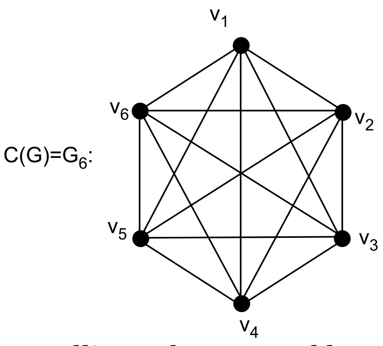


For the non-adjacent vertices $v_3$ and $v_6$ we have $d(v_3) + d(v_6) = 4 + 3 = 7 \geq n$. Therefore, there exists an edge between $v_3$ and $v_6$ . Let the super graph $G_5$ be.

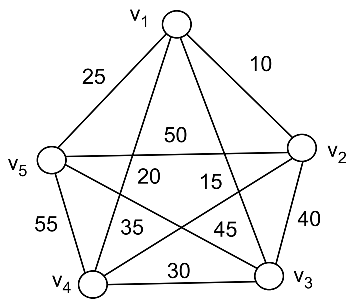
G$_5$:



For the non-adjacent vertices $v_4$ and $v_6$ we have $d(v_4) + d(v_6) = 4 + 4 = 8 \geq n$. Therefore, there exists an edge between $v_4$ and $v_6$ . Let the super graph be $G_6$ . In the above graph, there is no two non-adjacent vertices. Thus, $G_6$ is the final super graph. Therefore, the closure of the graph G is given as



**Example 4** *For the following travelling salesman problem, carry out the closest insertion algorithm.*



**Solution:** Given that the complete weighted graph G as

1. Choose the vertex $v_1$
2. Choose the vertex $v_2$, which is closest to $v_1$. So, $w_2 = v_1\, v_2\, v_1$
3. Choose the vertex $v_3$, which is close to $w_2$. So, $w_3 = v_1\, v_2\, v_3\, v_1$
4. Choose the vertex $v_4$, which is close to $w_3$. Hence, we have the following cases.

$$w_4 = v_1\, v_2\, v_3\, v_4\, v_1 \text{ or}$$
$$= v_1\, v_2\, v_4\, v_3\, v_1 \text{ or}$$
$$= v_1\, v_4\, v_2\, v_3\, v_1$$

Now length of $v_1\, v_2\, v_3\, v_4\, v_1 = 10 + 40 + 30 + 20 = 100$

Length of $v_1\, v_2\, v_4\, v_3\, v_1 = 10 + 45 + 30 + 15 = 100$

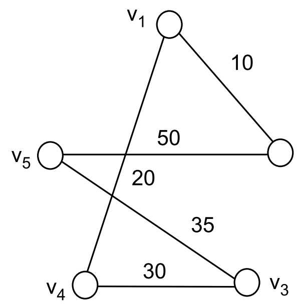
Length of $v_1\, v_4\, v_2\, v_3\, v_1 = 20 + 45 + 40 + 15 = 120$

Therefore, $w_4 = v_1\, v_2\, v_3\, v_4\, v_1$ is minimum.

5. Choose the vertex $v_5$, which is close to $w_4$. Hence, we have the following cases. The length of following cycles is given as below.

$$v_1\, v_2\, v_3\, v_4\, v_5\, v_1 = 10 + 40 + 30 + 55 + 25 = 160$$
$$v_1\, v_2\, v_3\, v_5\, v_4\, v_1 = 10 + 40 + 35 + 55 + 20 = 160$$
$$v_1\, v_2\, v_5\, v_3\, v_4\, v_1 = 10 + 50 + 35 + 30 + 20 = 145$$
$$v_1\, v_5\, v_2\, v_3\, v_4\, v_1 = 25 + 50 + 40 + 30 + 20 = 165$$

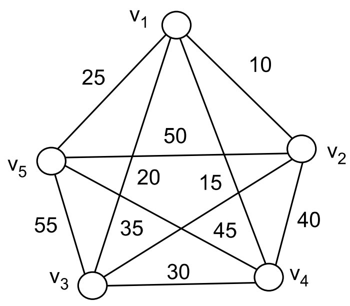
As all the vertices are included in the cycle, so the process terminates. Hence , the shortest Hamiltonian cycle is given as $v_1\, v_2\, v_5\, v_3\, v_4\, v_1$.



**Example 5** *For the travelling salesman problem given in example 4, carry out the two optimal algorithm.*

**Solution:** For the complete weighted graph G given above, the number of vertices $(n) = 5$. According to the two optimal algorithm we have the following steps:

1. Let $C = v_1, v_2, v_3, v_4, v_5, v_1$ be a Hamiltonian cycle.
   Therefore, we get
   $$w = w(v_1v_2) + w(v_2v_3) + w(v_3v_4) + w(v_4v_5) + w(v_5v_1)$$
   $$= 10 + 40 + 30 + 55 + 25 = 160$$
2. Set $i = 1$
3. Set $j = i + 2 = 3$
4. Set $C_{ij} = C_{13} = v_1\, v_3\, v_2\, v_4\, v_5\, v_1$
   $$w_{13} = w - w(v_1v_2) - w(v_3v_4) + w(v_1v_3) + w(v_2v_4)$$
   $$= 160 - 10 - 30 + 15 + 45 = 180$$
5. As $w_{13} \not< w$; Go to step 6.
6. Set $j = (j + 1) = 4$ and $4 \le n = 5$. Go to step 4.
4. Set $C_{ij} = C_{14} = v_1\, v_4\, v_3\, v_2\, v_5\, v_1$
   $$w_{14} = w - w(v_1v_2) - w(v_4v_5) + w(v_1v_4) + w(v_2v_5) = 165$$
5. As $w_{14} = 165 \not< 160 = w$; Go to step 6.
6. Set $j = (j + 1) = 5$ and $5 \le n = 5$. Go to step 4.
4. Set $C_{ij} = C_{15} = v_1\, v_5\, v_4\, v_3\, v_2\, v_1$
   $$w_{15} = w - w(v_1v_2) - w(v_5v_1) + w(v_1v_5) + w(v_2v_1) = 160$$
5. As $w_{15} = 160 \not< 160 = w$; Go to step 6
6. Set $j = (j + 1) = 6$ and $6 \not< n = 5$. Go to step 7 with $i = (i + 1) = 2$.
7. As $i = 2 \le (n - 2) = 3$, Go to step 3.
3. Set $j = (i + 2) = 2 + 2 = 4$
4. Set $C_{ij} = C_{24} = v_1\, v_2\, v_4\, v_3\, v_5\, v_1$
   $$W_{24} = w - w(v_2\, v_3) - w(v_4\, v_5) + w(v_2\, v_4) + w(v_3\, v_5) = 145$$
5. As $w_{24} = 145 < 160 = w$; go to step 1
   1. $C = C_{24} = v_1\, v_2\, v_4\, v_3\, v_5\, v_1$ with $w = w_{24} = 145$.
   After re-labeling the vertices we have
   $$C = C_{24} = v_1\, v_2\, v_3\, v_4\, v_5\, v_1.$$



2. Set $i = 1$

3. Set $j = (i + 2) = 3$

4. Set $C_{ij} = C_{13} = v_1\, v_3\, v_2\, v_4\, v_5\, v_1$

$$w_{13} = w - w(v_1\, v_2) - w(v_3\, v_4) + w(v_1\, v_3) + w(v_2\, v_4) = 165$$

5. As $w_{13} = 165 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 4$ and $4 \le n = 5$. Go to step 4

4. Set $C_{ij} = C_{14} = v_1\, v_4\, v_3\, v_2\, v_5\, v_1$

$$w_{14} = w - w(v_1\, v_2) - w(v_4\, v_5) + w(v_1\, v_4) + w(v_2\, v_5) = 165$$

5. As $w_{14} = 165 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 5$ and $5 \le n = 5$. Go to step 4

4. Set $C_{ij} = C_{15} = v_1\, v_5\, v_4\, v_3\, v_2\, v_1$

$$w_{15} = w - w(v_1\, v_2) - w(v_5\, v_1) + w(v_1\, v_5) + w(v_2\, v_1) = 145$$

5. As $w_{15} = 145 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 6 \not< n = 5$ with $i = (i + 1) = 2$.

7. As $i = 2 \le (n - 2) = 3$, Go to step 3

3. Set $j = (i + 2) = 2 + 2 = 4$

4. Set $C_{ij} = C_{24} = v_1\, v_2\, v_4\, v_3\, v_5\, v_1$

$$w_{24} = w - w(v_2\, v_3) - w(v_4\, v_5) + w(v_2\, v_4) + w(v_3\, v_5) = 160$$

5. As $w_{24} = 160 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 5 \le 5 = n$, go to step 4

4. Set $C_{ij} = C_{25} = v_1\, v_2\, v_5\, v_4\, v_3\, v_1$

$$w_{25} = w - w(v_2\, v_3) - w(v_5\, v_1) + w(v_2\, v_5) + w(v_3\, v_1) = 145$$

5. As $w_{25} = 145 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 6 \not< n = 5$ with $i = (i + 1) = 3$. go to step 7

7. As $i = 3 \le (n - 2) = 3$, go to step 3

3. Set $j = (i + 2) = 5$

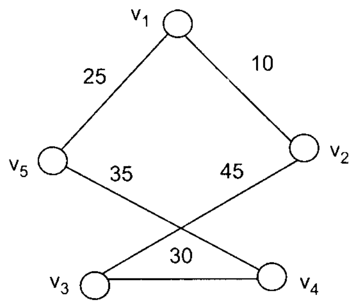
4. Set $C_{ij} = C_{35} = v_1\, v_2\, v_3\, v_5\, v_4\, v_1$

$$w_{35} = w - w(v_3\, v_4) - w(v_5\, v_1) + w(v_3\, v_5) + w(v_4\, v_1) = 160$$

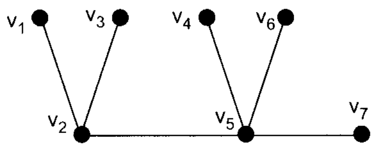
5. As $w_{35} = 160 \not< 145 = w$; go to step 6

6. Set $j = (j + 1) = 6 \not< n = 5$ with $i = (i + 1) = 4$, go to step 7

7. As $i = 4 \not< (n - 2) = 3$, therefore the process terminates. Hence, the minimum Hamiltonian path is given as $v_1\, v_2\, v_3\, v_4\, v_5\, v_1$. The path for travelling salesman is given below.



**Example 6** *Find the eccentricity of all vertices, radius, diameter and centre of the graph given below. It is given that the distance between any two adjacent vertices is 1.*



**Solution:** In the graph given above V = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. It is also given that length of each edge is 1. Now,

$d(v_1, v_2) = 1$; $d(v_1, v_3) = 2$; $d(v_1, v_4) = 3$;
$d(v_1, v_5) = 2$; $d(v_1, v_6) = 3$; $d(v_1, v_7) = 3$;

Therefore, $e(v_1) = \text{Max } \{1, 2, 3\} = 3$.

Similarly, we get

$d(v_2, v_1) = 1$; $d(v_2, v_3) = 1$; $d(v_2, v_4) = 2$;
$d(v_2, v_5) = 1$; $d(v_2, v_6) = 2$; $d(v_2, v_7) = 2$;

Therefore, $e(v_2) = \text{Max } \{1, 2\} = 2$.

Proceeding in this way we get

$e(v_3) = 3; e(v_4) = 3; \ e(v_5) = 2; \ e(v_6) = 3; \ e(v_7) = 3.$

Now, radius = rad (G) = Min $\{e(v): v \in \text{V}\}$ = Min (2, 3) = 2.

Diameter = diam (G) = Max $\{e(v): v \in \text{V}\}$ = Max (2, 3) = 3.

So, the central points are $v_2$, $v_5$ and centre $\{v_2, v_5\}$.

**Example 7** *Let T be a tree of order p and size q having $p_i$ vertices of degree i (i = 1, 2, 3, ...).*

*Let* $\sum_i p_i = p$ *and* $\sum_i ip_i = 2q = 2(p-1)$.

*Show that* $p_1 = p_3 + 2p_4 + 3p_5 + 4p_6 + \ldots + 2$.

**Solution:** Given that T is a tree of order $p$ and size $q$. It is also given that

$$\sum_i p_i = p \text{ and } \sum_i ip_i = 2(p-1)$$

*i.e.,* $p_1 + 2p_2 + 3p_3 + 4p_4 + \ldots = 2p - 2$

*i.e.,* $p_1 + 2p_2 + 3p_3 + 4p_4 + \ldots = 2\sum_i p_i - 2$

*i.e.*, $p_1 + 2p_2 + 3p_3 + 4p_4 + ... = 2(p_1 + p_2 + p_3 + ...) - 2$

*i.e.*, $$p_1 = p_3 + 2p_4 + 3p_5 + 4p_6 + ... + 2.$$

**Example 8** *If T is a binary tree of height h and order p, then*

$$(h + 1) \leq p \leq 2^{(h+1)} - 1$$

**Solution:** Let $p_k$ denotes the number of vertices of T at level $k$ for $0 \leq k \leq h$.

Therefore, we get

$$\sum_{k=0}^{h} p_k = p$$

Since $p_k \geq 1$ for each $k$, and $p_k \leq 2p_{(k-1)}$ for $1 \leq k \leq h$, it follows, inductively, that $p_k \leq 2^k$. Again,

$$\sum_{k=0}^{h} 2^k = 1 + 2 + 2^2 + 2^3 + ... + 2^h = 2^{h+1} - 1$$

Again, $$\sum_{k=0}^{h} p_k \leq \sum_{k=0}^{h} 2^k = 2^{h+1} - 1$$

*i.e.*, $$p \leq 2^{h+1} - 1 \qquad ...(i)$$

Also, $$\sum_{k=0}^{h} 1 \leq \sum_{k=0}^{h} p_k = p$$

*i.e.*, $$(h + 1) \leq p \qquad ...(ii)$$

On combining equations (*i*) and (*ii*), we get

$$(h + 1) \leq p \leq 2^{h+1} - 1.$$

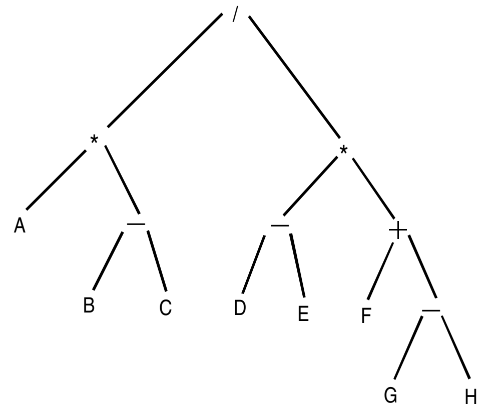
**Example 9** *Construct the binary tree for the arithmetic expression*

*(A(B – C)) / ((D – E)(F + G – H)).*

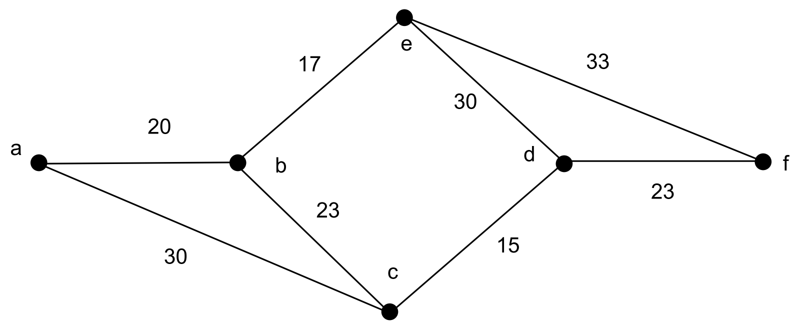
**Solution:** Given arithmetic expression is

(A(B – C))/((D – E)(F + G – H)).

The binary tree corresponding to the above expression is given below.



**Example 10** *For the graph G shown below, use Dijkstra's algorithm to compute the shortest path between a and f.*



**Solution:** In the above graph G, the source vertex is

$v_s = a$ and $v_t = f$. Set $\lambda(a) = 0$

and $\lambda(b) = \lambda(c) = \lambda(d) = \lambda(e) = \lambda(f) = \infty$. T = V = $\{a, b, c, d, e, f\}$.

Hence, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| T | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |

Now, $u = a$ as $\lambda(u) = \lambda(a) = 0$

which is minimum.

The edges incident on $u = a$ are $ab$ and $ac$.

Therefore,

$$\lambda(b) = \text{Min}\,[\lambda(b), \lambda(a) + w(ab)]$$
$$= \text{Min}\,[\infty, 20] = 20.$$
$$\lambda(c) = \text{Min}\,[\lambda(c), \lambda(a) + w(ac)]$$
$$= \text{Min}\,[\infty, 30] = 30.$$

Again, T = T– $\{u = a\} = \{b, c, d, e, f\}$.

Therefore, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 20 | 30 | $\infty$ | $\infty$ | $\infty$ |
| T |  | $b$ | $c$ | $d$ | $e$ | $f$ |

Now, $u = b$ as $\lambda(u) = \lambda(b) = 20$

which is minimum.

The edges incident with $u = b$ are $bc$ and $be$. Therefore,

$$\lambda(c) = \text{Min}\,[\lambda(c), \lambda(b) + w(bc)]$$
$$= \text{Min}\,[30, 43] = 30.$$
$$\lambda(e) = \text{Min}\,[\lambda(e), \lambda(b) + w(be)]$$
$$= \text{Min}\,[\infty, 37] = 37.$$

Again, T = T – $\{u = b\}$ =$\{c, d, e, f\}$.

Thus, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 20 | 30 | $\infty$ | 37 | $\infty$ |
| T | | | $c$ | $d$ | $e$ | $f$ |

Now, $u = c$ as $\lambda(u) = \lambda(c) = 30$ which is minimum.

The edges incident with $u = c$ is $cd$.

Therefore, $\lambda(d) = \text{Min}\,[\lambda(d), \lambda(c) + w(cd)]$
$= \text{Min}\,[\infty, 45] = 45.$

Again, $T = T - \{u = c\} = \{d, e, f\}$.

Thus, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 20 | 30 | 45 | 37 | $\infty$ |
| T | | | | $d$ | $e$ | $f$ |

Now, $u = e$ as $\lambda(u) = \lambda(e) = 37$ which is minimum.

The edges incident with $u = e$ are $ed$ and $ef$.

Therefore, $\lambda(d) = \text{Min}\,[\lambda(d), \lambda(e) + w(ed)]$
$= \text{Min}\,[45, 67] = 45.$

$\lambda(f) = \text{Min}\,[\lambda(f), \lambda(e) + w(ef)]$
$= \text{Min}\,[\infty, 70] = 70.$

Again, $T = T - \{u = e\} = \{d, f\}$.

Thus, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 20 | 30 | 45 | 37 | 70 |
| T | | | | $d$ | | $f$ |

Now, $u = d$ as $\lambda(u) = \lambda(d) = 45$ which is minimum.

The edges incident with $u = d$ is $df$. Therefore,

$\lambda(f) = \text{Min}\,[\lambda(f), \lambda(d) + w(df)]$
$= \text{Min}\,[70, 68] = 68.$

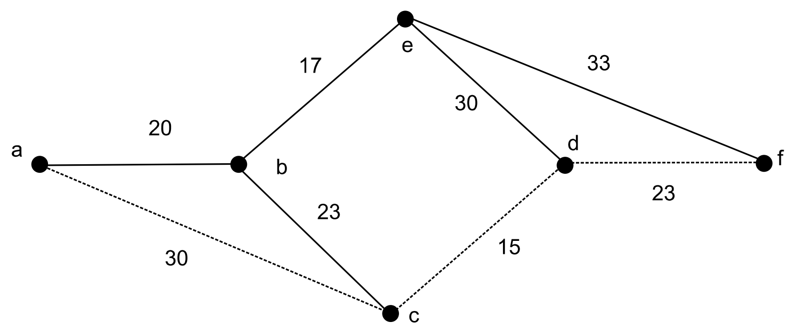
Again, $T = T - \{u = d\} = \{f\}$.

Thus, we have the following table

| Vertex | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| $\lambda(v)$ | 0 | 20 | 30 | 45 | 37 | 68 |
| T | | | | | | $f$ |

Now, $u = f$ and $f$ is the terminating node, so the process terminates.

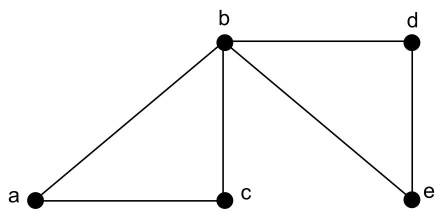
Hence, the shortest distances from $a$ to $b$, $c$, $d$, $e$ and $f$ are 20, 30, 45, 37, 68 respectively. The shortest distance between $a$ and $f$ is given in the figure further.



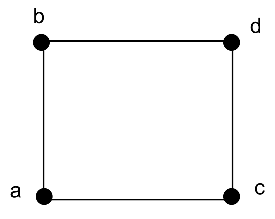
**Example 11** *Construct the following graphs.*

*(a) Eulerian but not Hamiltonian* *(b) Hamiltonian but not Eulerian*
*(c) Neither Eulerian nor Hamiltonian* *(d) Eulerian and Hamiltonian*

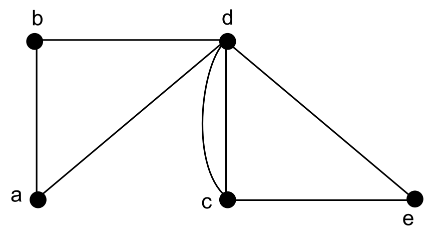
**Solution:** The different graphs are given below.



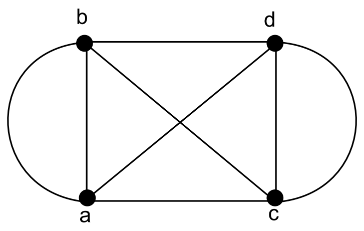
Eulerian but not Hamiltonian



Hamiltonian but not Eulerian

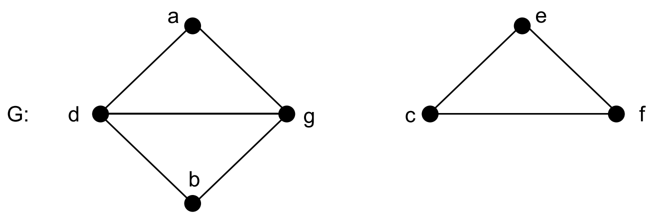


Neither Eulerian nor Hamiltonian

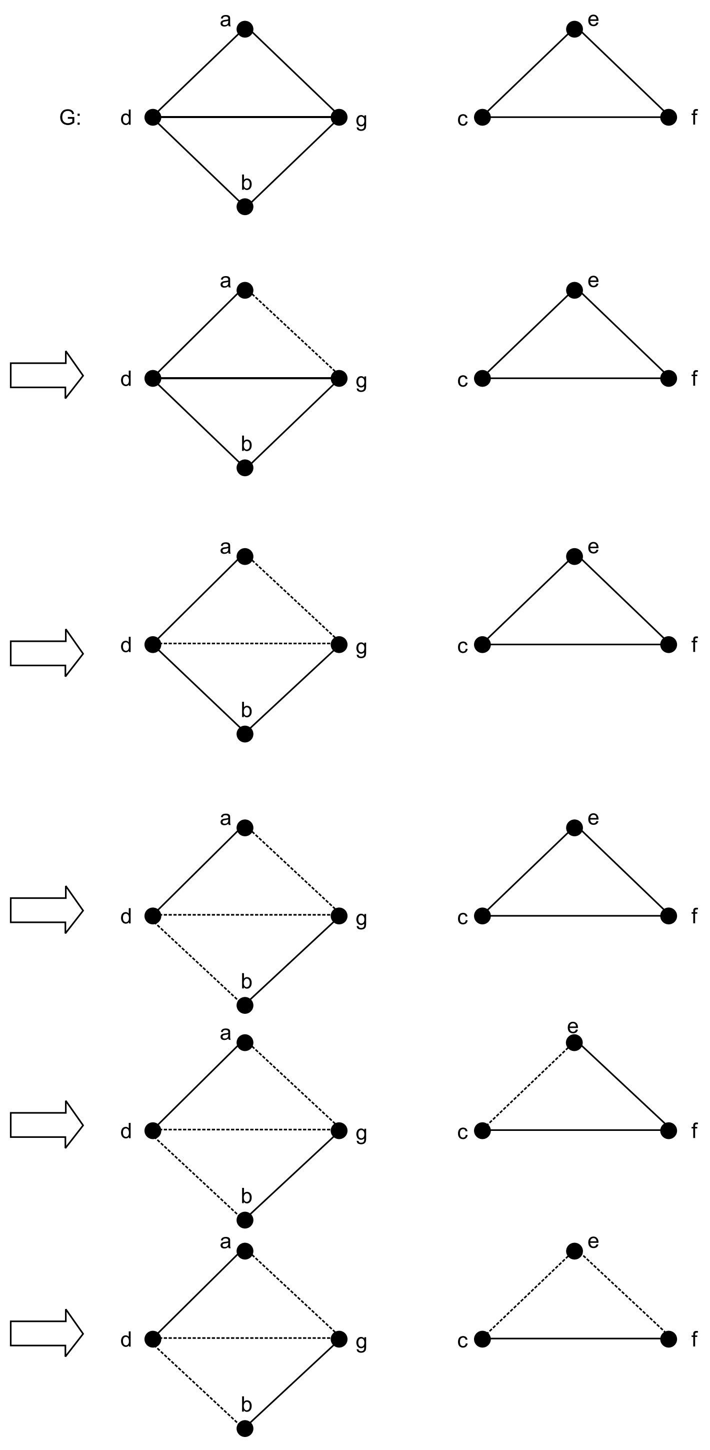


Hamiltonian and Eulerian

**Example 12** *For the graph G shown below, find the depth first search forest.*

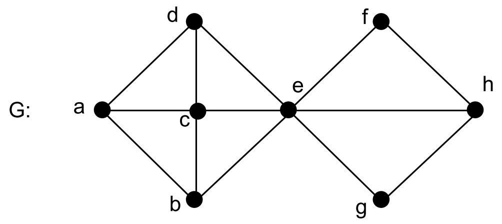


**Solution:** Let us consider the source vertex as '*a*' in the above graph G. On using the DFS technique, the order in which the vertices are being visited is described below by the sequence of graphs.

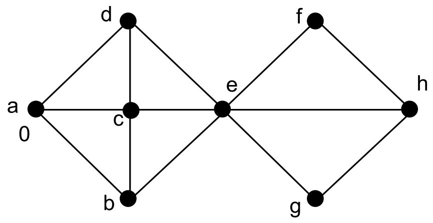


Therefore, the dotted graph shown above is the depth first forest T of the graph G. Besides that there could be several depth first forest from the same vertex '*a*'. This indicates that the depth first forest is not unique.

**Example 13** *For the graph G shown below, find the breadth first search tree.*



**Solution:** Consider the graph G given above. Now we have to find out the shortest path from the source vertex *a* to the vertex *h*. On using the BFS technique, we get the following stages.



(Label (*a*) = 0 and set *i* = 0 )

In the above figure the adjacent vertices of *a* are *b*, *c* and *d*. Therefore we get label

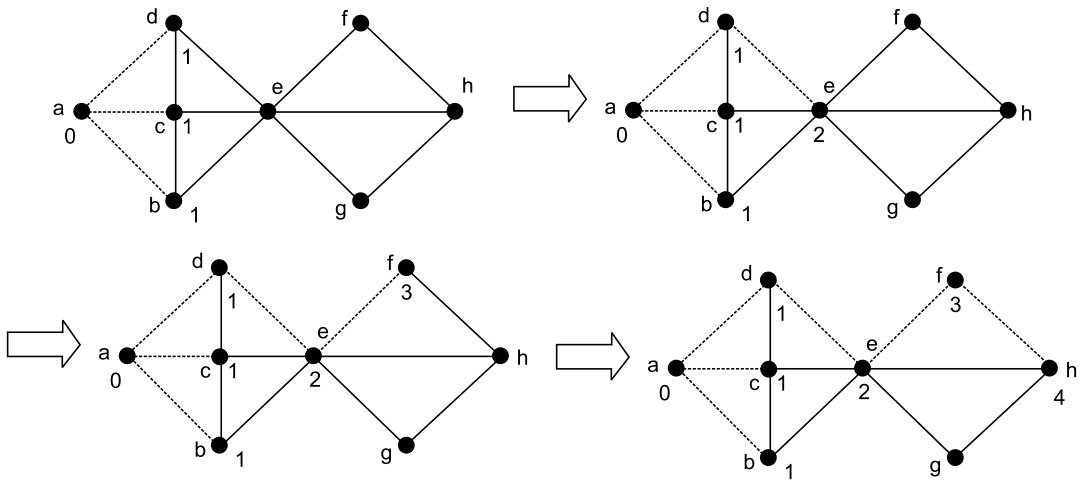
label (*b*) = *i* + 1 = 0 + 1 = 1;

label (*c*) = *i* + 1 = 0 + 1 = 1 and label (*d*) = *i* + 1 = 0 + 1 = 1.

Similarly, the adjacent vertex of *d* is *e*.

Therefore we get label (*e*) = *i* + 1 = 1 + 1 = 2.

Therefore, we have



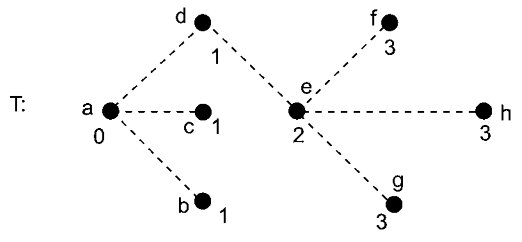
In the above figure, the adjacent vertex of *e* are *f*, *g* and *h*. Therefore we get,

label (*f*) = *i* + 1 = 2 + 1 = 3.

label (*g*) = *i* + 1 = 2 + 1 = 3.

label (*h*) = *i* + 1 = 2 + 1 = 3.

Therefore, the breadth first search tree is given as below.



## EXERCISES

**1.** With reference to the given tree T find the followings.

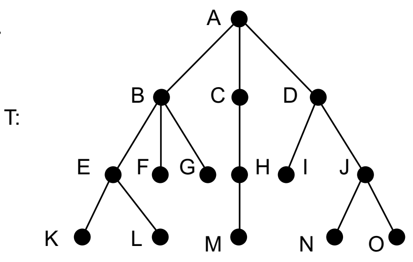
(*a*) Height of the tree

(*b*) Degree of the tree

(*c*) Longest path of the tree

(*d*) Level (L); Level (H); Level (N)

(*e*) Parent (M); Sibling (B); Child (D)



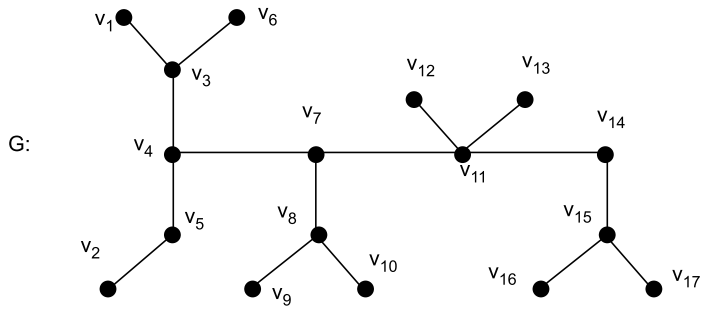
**2.** (*a*) Draw all trees of order 5

(*b*) Draw all trees of order 7 and $\Delta(T) \geq 4$, where $\Delta(T)$ represents maximum degree of tree T.

**3.** In a binary tree of height $h$, there are at most $2^{h-1}$ leaf nodes.

**4.** If T is a binary tree of height $h$ and order $p$, then $h \geq \lceil g\,((p+1)/2) \rceil$. The equality holds if T is a balanced complete binary tree.

**5.** Find the eccentricity of all vertices, radius, diameter and centre of the graph G given below. It is given that the distance between any two adjacent vertices is 1.



**6.** Construct the following graphs.

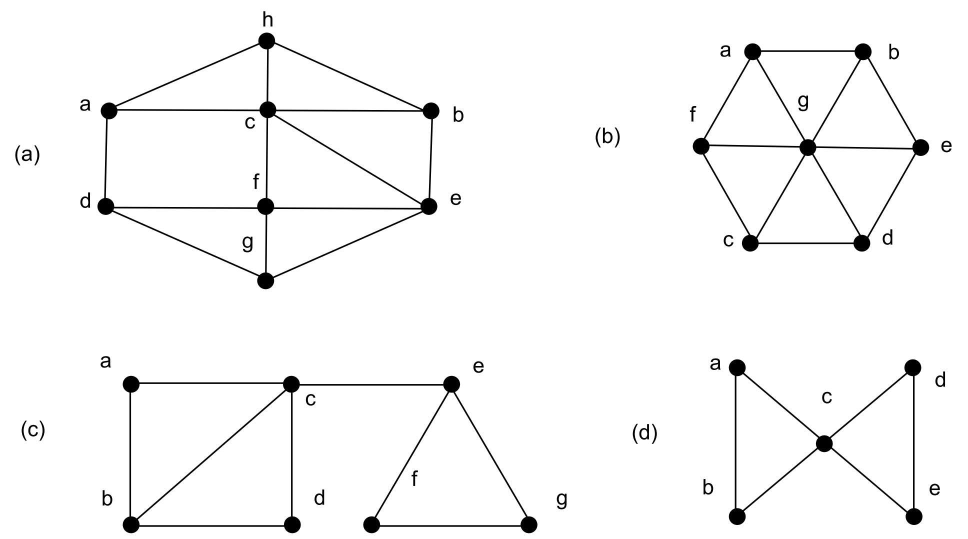
(*a*) Eulerian but not Hamiltonian

(*b*) Hamiltonian but not Eulerian

(*c*) Neither Eulerian nor Hamiltonian

(*d*) Eulerian and Hamiltonian.

**7.** For the graph G shown below, find the depth-first search tree.

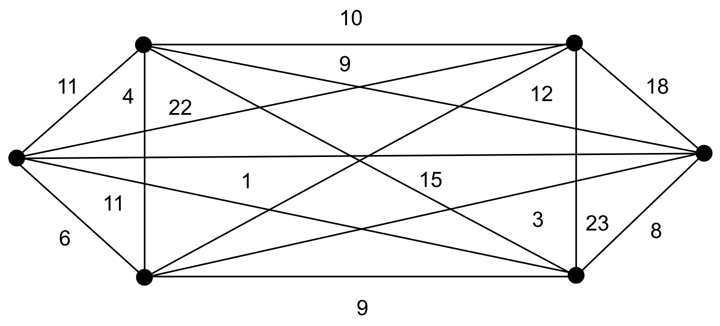


**8.** For the graphs given on No. 7, find the breadth first search tree.

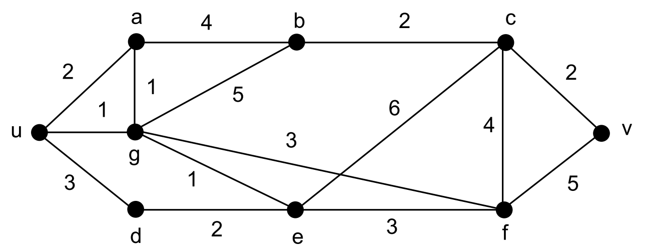
**9.** Solve the travelling salesman problem for the complete weighted graph G given below by using

(*a*) Closest insertion algorithm and

(*b*) Two optimal algorithm.

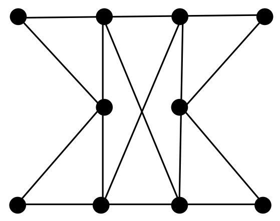


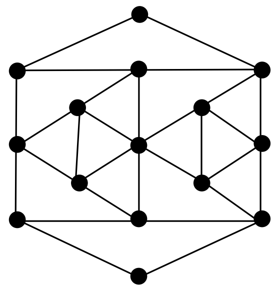
**10.** Let G be the weighted graph shown below. Use Dijkstra's algorithm to compute the shortest distance between *u* and *v*.



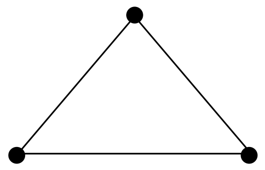
**11.** Determine which of the graphs given below are Euler graph by using the following algorithms.

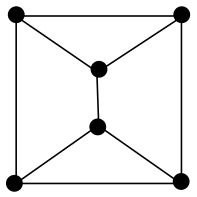
(*a*) Fleury's algorithm and (*b*) Hierholzer's algorithm

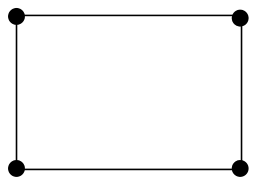
(a) 

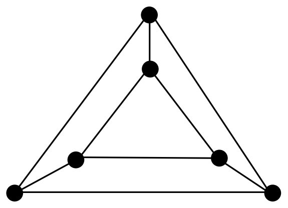
(b) 

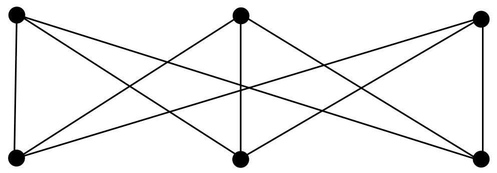
**12.** Find the closure graph C(G) for the graphs shown below.

(a) 

(b) 

(c) 

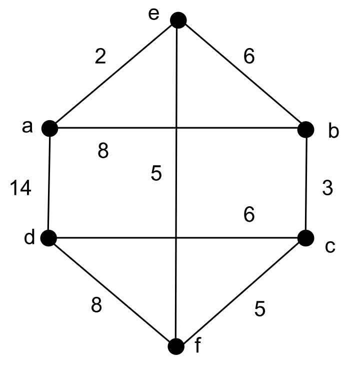
(d) 

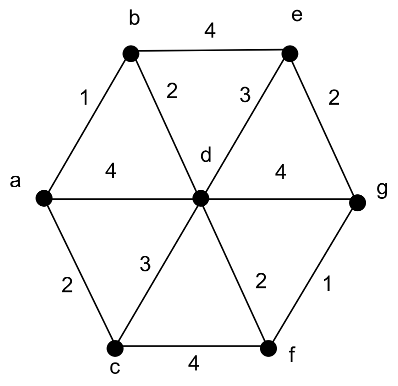
(e) 

**13.** Find the binary tree representation of the followings.

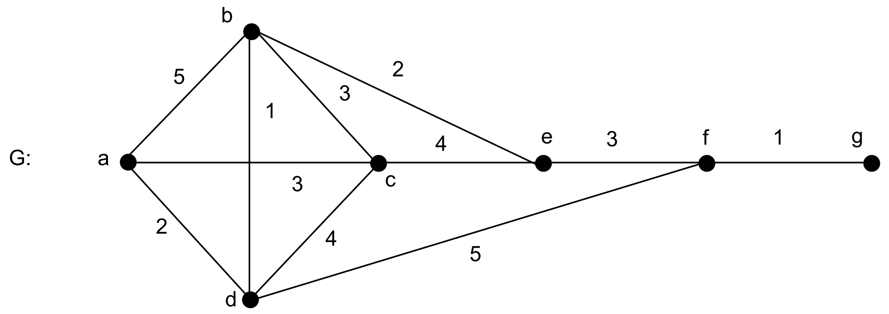
(*a*) $(4x + 2)(2x + xy)$ (*b*) $(x + 3y) - ((5x + y)/4)$

**14.** Let G be a connected weighted graph. Use Dijkstra's algorithms to find the length of shortest paths from the vertex $a$ to each of the other vertices.

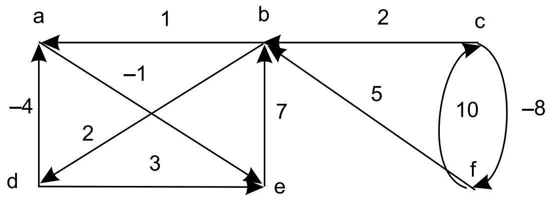
(a) 

(b) 

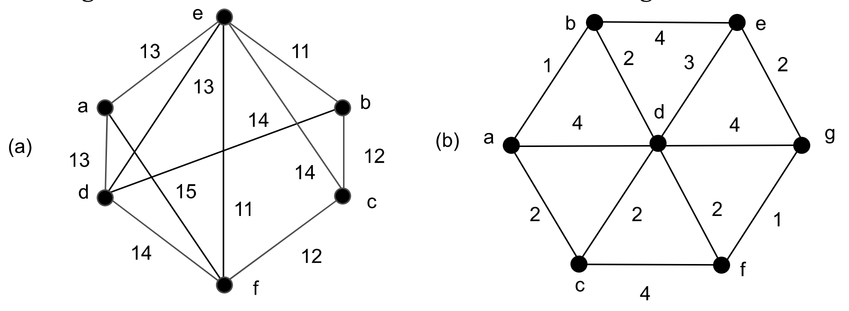
**15.** Apply Dijkstra's algorithm to the weighted graph G below to find the shortest distance for each vertex from the source vertex $a$.

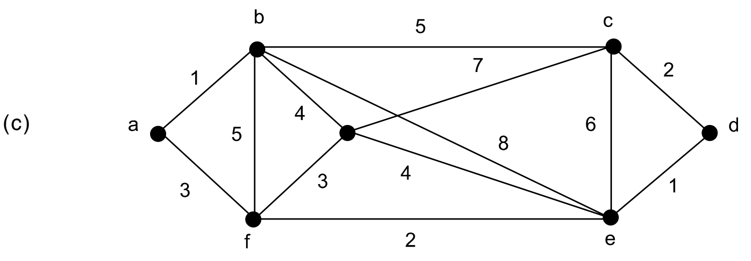


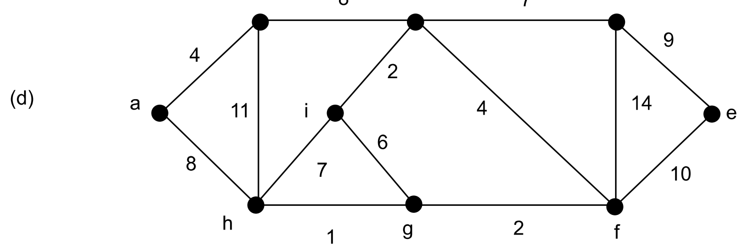
**16.** Use Floyd-Warshall algorithm on the weighted, directed graph G shown below to find out shortest path between any pair of vertices. Show the matrix D($k$) that results for each iteration.



**17.** Find the minimum spanning tree of the graphs shown below by using
($a$) Prim's algorithm and ($b$) Kruskal's algorithm.

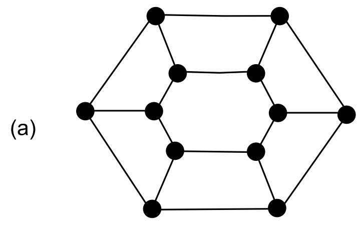


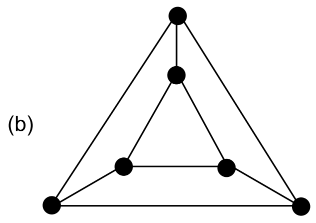




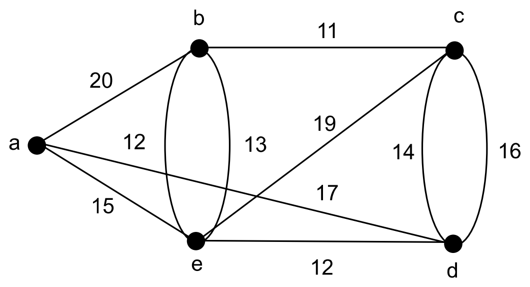
**18.** Find a maximal spanning tree for each of the graphs of No. 17. using either Prim's algorithm or Kruskal's algorithm. [**Hint:** To get the maximal spanning tree replace the weight of each edge of the graph by M – $w(e)$, where M is any number greater than the weight $w(e)$ of every edge $e$ of the graph. Then apply Prim's algorithm or Kruskal's algorithm. The corresponding spanning tree in the original weighted graph is a maximal spanning tree.]

**19.** Find the closure graph C(G) of the following graphs.





**20.** Let G be a connected weighted graph. Use Floyd-Warshall algorithm to find the length of shortest path between any pair of vertices.


b
c
a
e
d
11
20
12
13
19
14
16
17
15
12