

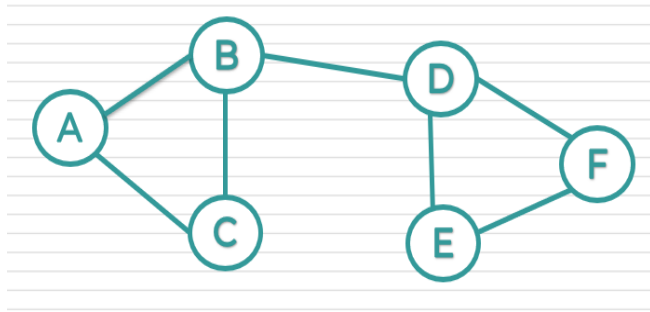
GRAPHS

A graph G is a set of vertices (V) and set of edges (E)

$$G=(V,E)$$

$V(G)$ =Vertices of graph G

$E(G)$ =Edges of graph G



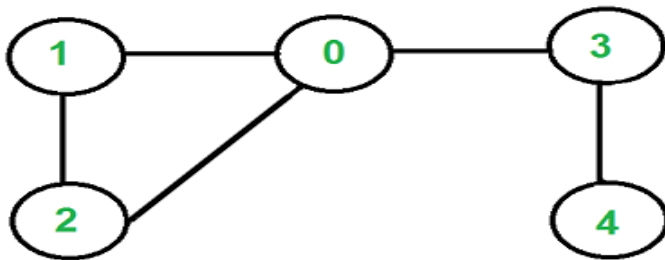
$$V(G_1)=\{A,B,C,D,E,F\}$$

Types of Graph

1. Undirected Graph

A graph containing unordered pair of vertices is called an undirected graph.

In an undirected graph, pair of vertices $(1,0)$ and $(0,1)$ represent the same edge.

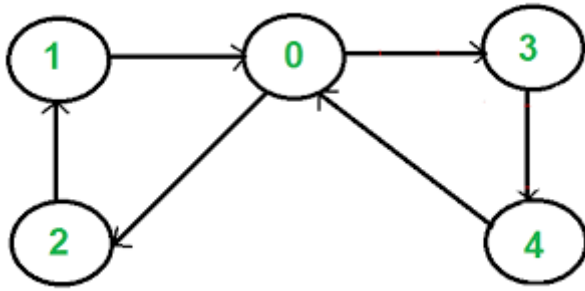


2. Directed Graph

A graph containing ordered pair of vertices is called a direct graph.

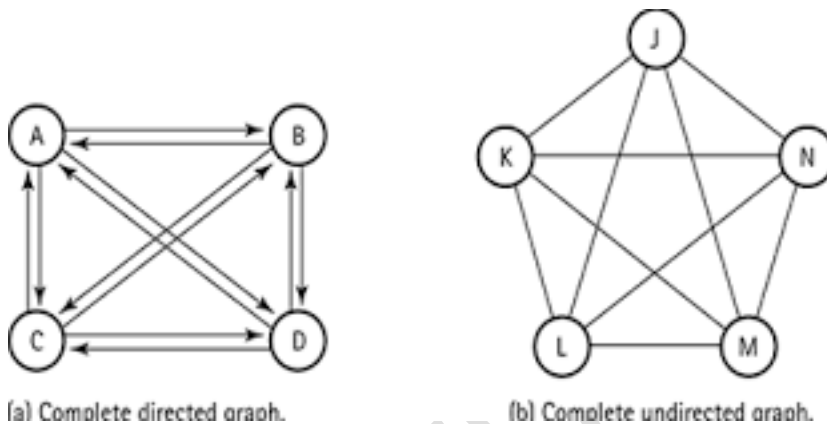
If an edge is represented using a pair of vertices $(1,0)$ then the edge is said to be directed from 1 to 0

1 is called predecessor of 0 and 0 is successor



3. Complete Graph

An undirected graph, in which every vertex has an edge to all other vertices is called a complete graph.

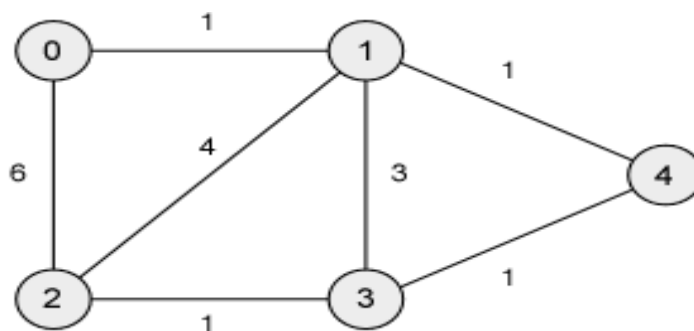


4. Weighted Graph

A weighted graph is a graph in which edges are assigned some value. Most of the physical situation shown using weighted graph.

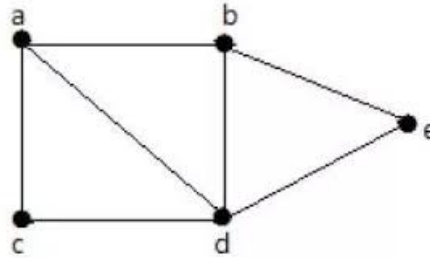
A weighted graph is a graph in which edges are assigned some value.

The weight will denote the distance between two connected cities using highway



Cycle

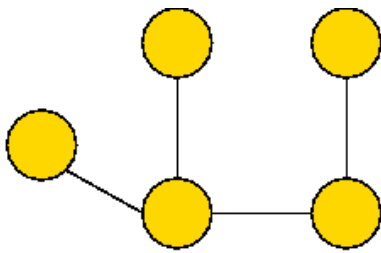
A cycle is a simple path that begins and ends at the same vertex.



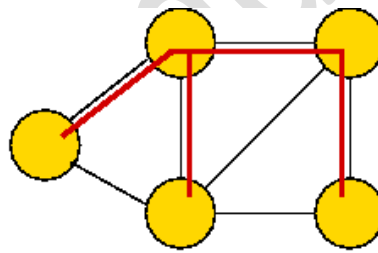
Cost : The cost of a graph is the sum of the cost of the edges in the weighted graph.

TREE

Tree is connected graph without any cycle,



A TREE



Spanning tree of a graph

Degree of Vertex:-

The total number of edges linked to a vertex is called its degree.

The indegree of a vertex is the total number of edges coming to that node .

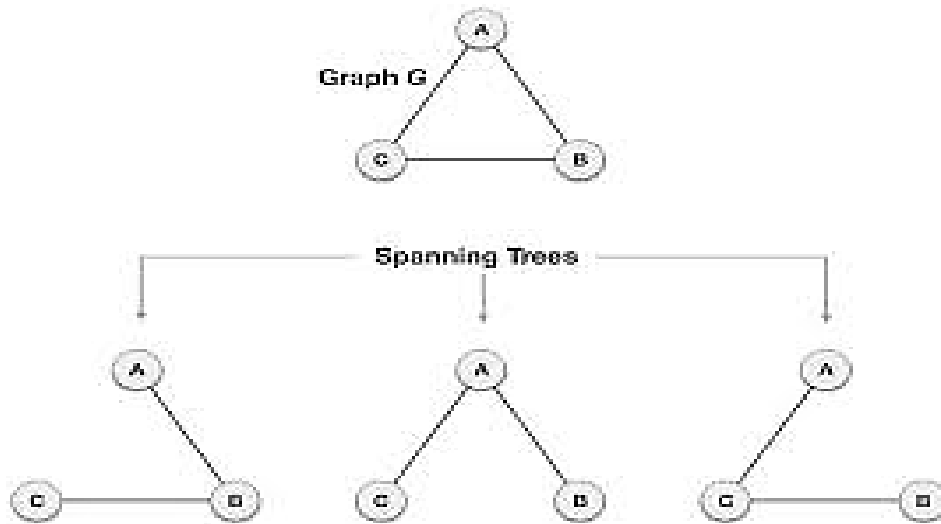
The outdegree of a node is the total number of edges going out from that node.

A vertex, which has only outgoing edges and no incoming edges, is called source

Spanning tree

A spanning tree of graph $G=(V,E)$ is subgraph of G having all vertices of G and no cycle in it.

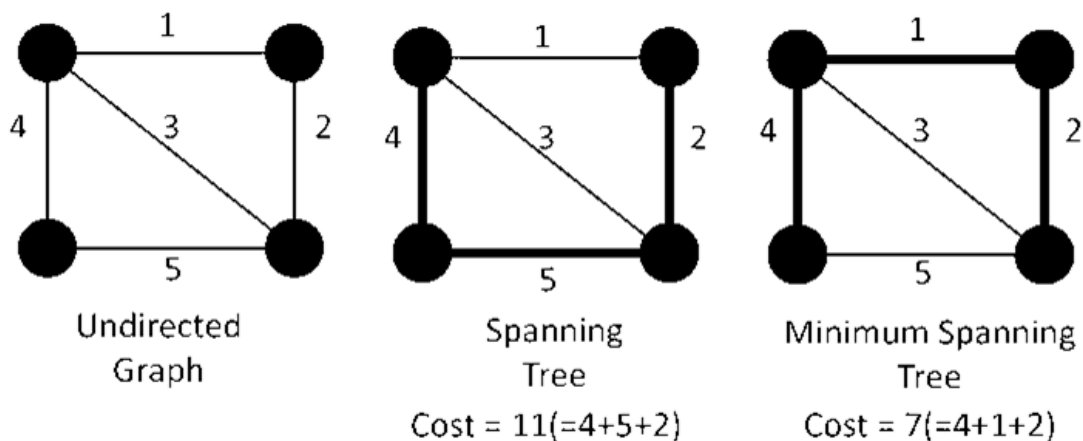
A spanning tree of a group $G=(V,E)$ is called a minimal cost spanning tree or simply the minimal spanning tree of G if its cost is minimum
If graph G is not connected then there is no spanning trees



Minimal Spanning Tree

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees.

Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.



Representation of graphs

Graph can be represented using two ways

1. Adjacency matrix

A two dimensional matrix can be used to store a graph.

A graph is represented using a square matrix.

If graph $G=(V,E)$ where vertices $V=\{0,1,2,3...\}$ can be represented using a two dimensional array.

e.g.

`int adj[20][20];`

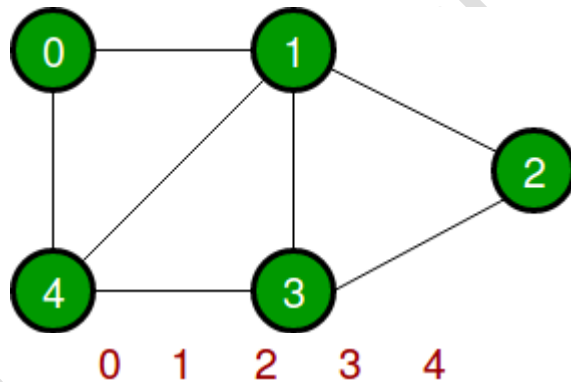
can be used to store a graph with 20 vertices.

If $\text{adj}[i][j]=1$, indicates presence of edge between two vertices i and j .

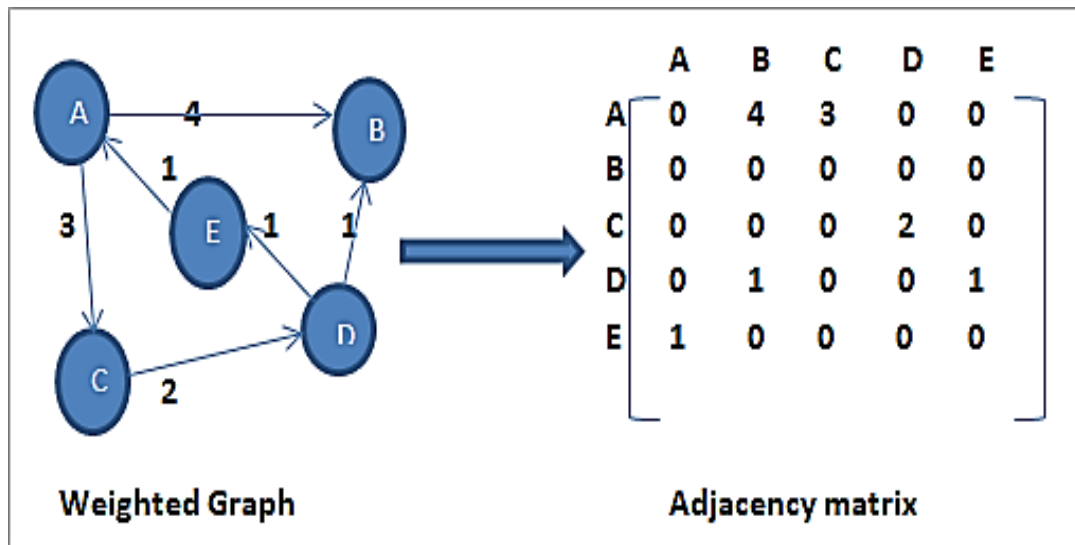
$\text{adj}[i][j]=0$, indicates absence of edge between two vertices i and j .

Adjacency matrix of an undirected graph is always a symmetric matrix

Adjacency matrix of a directed graph is never symmetric



| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 0 |

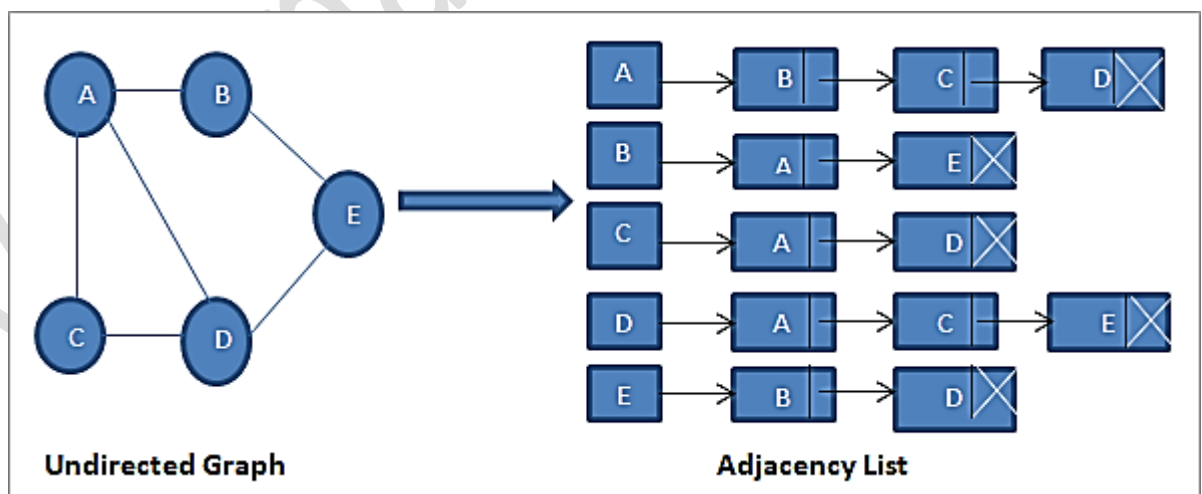


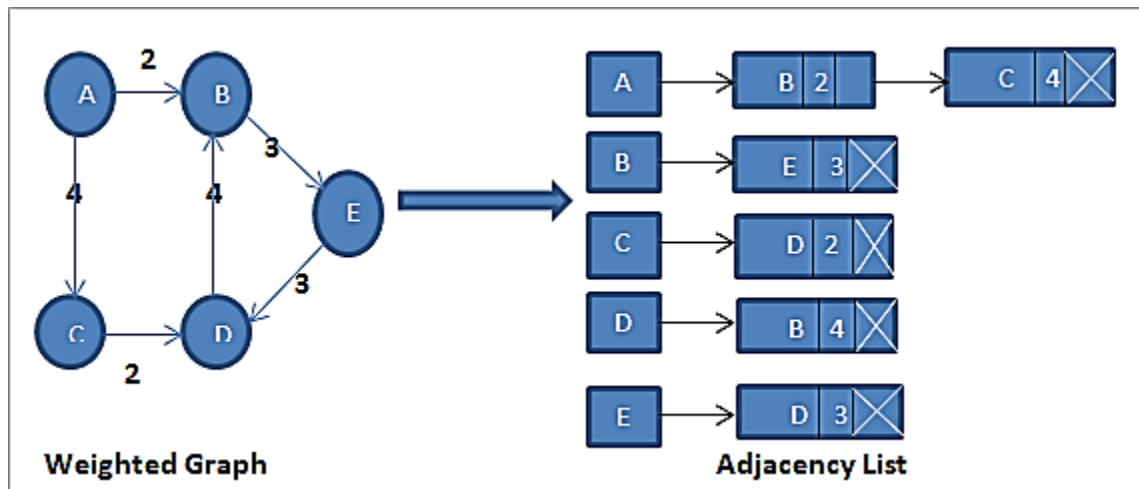
Adjacency List

A graph can be represented using a linked list. For each vertex a list of adjacent vertices is maintained using a linked list.

The adjacency list representation maintains each node of the graph and a link to the nodes that are adjacent to this node.

When we traverse all the adjacent nodes, we set the next pointer to null at the end of the list.





Traversal of graph

Graph traversal means visiting each node of graph exactly once.

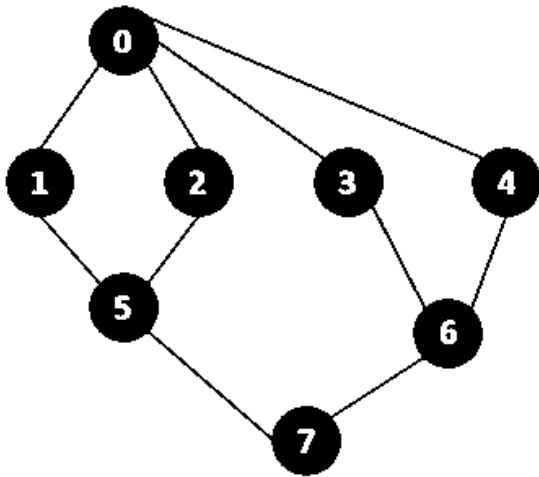
We need standard techniques to keep things uniform.

Different techniques used for graph traversal are

1. Depth First Search(DFS)
2. Breadth First Search(BFS)

Depth First Search(DFS)

- This algorithm traverses the tree depth-wise.
- It is like preorder traversal of a tree.
- In this method all vertices are stored in stack and each vertex of graph is visited or explored once.
- The newest vertex(added last) in the stack is explored first.
- Traversal can start from any vertex say v.
- V is visited then all vertices adjacent to v are visited recursively using dfs.
- Since graph can have cycle, we must avoid revisiting a node.
- When we visit vertex v, we marks it visited, a node that has already been marked as visited should not be selected for traversal.



Algorithm

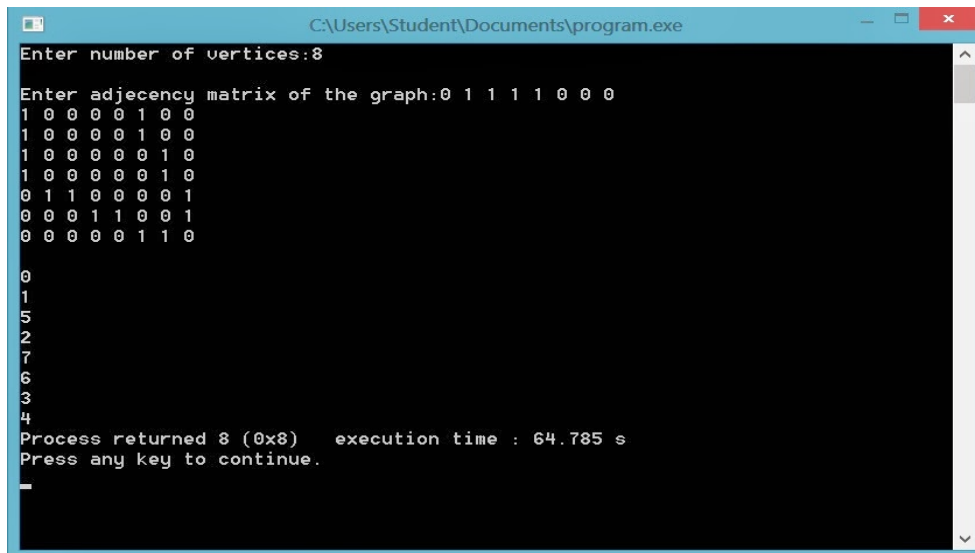
1. Initialize all the vertices.
2. Initialize array of visited[] to false (0).
3. Push starting vertex onto stack.
4. Repeat steps 5 and 6 until stack is empty.
5. Pop the top vertex from stack.
6. Push onto stack all the neighbors of popped vertex
7. End.

Depth First Search (DFS) Program in C

```
#include<stdio.h>
#include<conio.h>
void DFS(int);
int G[10][10];
int visited[10],n; //n is no of vertices and graph is sorted in array
void main()
{
    int i,j;
    printf("Enter number of vertices:");
    scanf("%d",&n);
    //read the adjacency matrix
    printf("\nEnter adjacency matrix of the graph:");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
```



```
        {
            scanf("%d",&G[i][j]);
        }
    }
    //visited is initialized to zero
    for(i=0;i<n;i++)
    {
        visited[i]=0;
    }
    BFS(0);
}
void BFS(int i)
{
    int j;
    init(&q);
    enqueue(&q, i);
    printf("\n %d",i);
    visited[i]=1;
    while(!empty(&q))
    {
        V=dequeue(&q);
        For(j=0;j<n;i++)
        {
            if(!visited[i]==0 && G[i][j]==1)
            {
                Enqueue(&q,i);
                Visited[i]=1;
                printf(" Visited %d",i);
            }
        }
    }
}
```



```

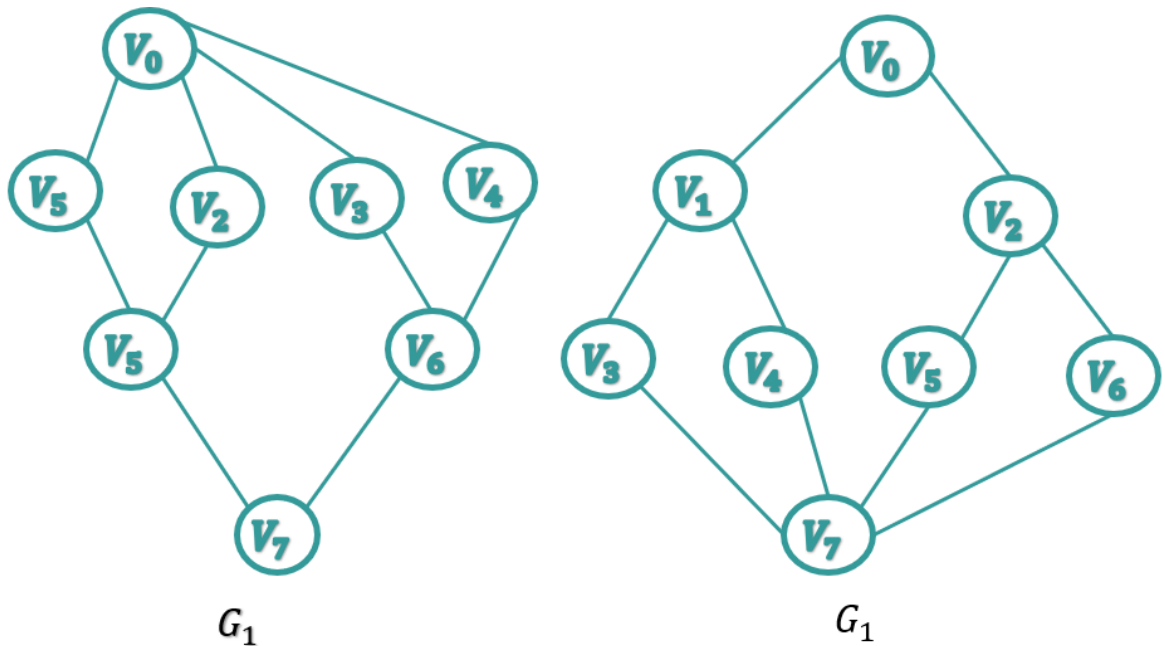
C:\Users\Student\Documents\program.exe
Enter number of vertices:8
Enter adjacency matrix of the graph:0 1 1 1 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 1 1 0 0 1
0 0 0 0 0 1 1 0

0
1
5
2
7
6
3
4
Process returned 8 (0x8)   execution time : 64.785 s
Press any key to continue.

```

Breadth First Search(BFS)

- It is another popular approach used for visiting the vertices of a graph.
- This method starts from a give V_0 . V_0 is marketed as visited.
- All vertices adjacent to V_0 are visited next.
- The algorithm for BFS has to maintain a list of vertices which have been visited but not explored for adjacent vertices.
- The vertices which have been visited but not explored for adjacent vertices can be stored in queue.
- Initially the queue contains the starting vertex.
- In every iteration, a vertex is removed from the queue and its adjacent vertices which are not visited as yet are added to the queue.
- The algorithm terminates when the queue become empty.

**Algorithm**

1. Initialize all the vertex.
2. Initialize array of visited [] to false (0)
3. Enqueue starting vertex in queue.
4. Mark it as visited [] to true (1)
5. Dequeue from queue.
6. Add to rear of queue all neighbors of deleted node that are not visited.
7. If queue is not empty go to step5.
8. End.

```
#include<stdio.h>
#include<conio.h>
typedef struct Queue
{
    Int data[max];
    Int F,R;
}Queue;
Void enqueue(Queue *q,int x);
Void Init(Queue *q);
int dequeue(Queue *q);
int empty((Queue *q);
void BFS(int);
```

```
int G[10][10];
int visited[10],n; //n is no of vertices and graph is sorted in array
void main()
{
    int i,j,v;
    printf("Enter number of vertices:");
    scanf("%d",&n);
    //read the adjacency matrix
    printf("\nEnter adjacency matrix of the graph:");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&G[i][j]);
        }
    }
    //visited is initialized to zero
    for(i=0;i<n;i++)
    {
        visited[i]=0;
    }
    Printf("Enter the starting vertex\n");
    Scanf("%d",&v);
    BFS(v);
    getch();
}
Void BFS(int v)
{
    Queue q;
    Init(&q);
    Enqueue(&q, v);
    printf("%d", v);
    visited [v]=1;
    while(!empty(&q))
    {
        V=dequeue(&q);
        For ( i=0 ; i<n ; i++)
```

```
        {
            If(visited[i]==0 && G[V][i]==1)
            {
                enqueue( &q, i);
                visited[i]=1;
                printf("%d",i);
            }
        }
    }
}
```

Important questions

What is graph explain methods to represent graph [May, DEC-18 5M]

Explain Graph Traversal Techniques [DEC17,MAY-18 10M]

Give c function for BFS traversal in a graph? explain with example?
[10M DEC -18]

Define minimum spanning tree ? list the technique to compute
minimum spanning tree [3M MAY-18]

Explain BFS and DFS algorithm with example [10M MAY-18]

Explain kruskal's algorithm with example [10M May-18]

Write a short note on prims algorithm and Dijkstra algorithm [10M
May-18]

Define graph? List the types of graph with example? [3M DEC-18]

What is minimal spanning tree ? Draw MST using kruskal's and prims
algorithm and find out cost. [10M DEC-18]

Explain BFS and DFS algorithm with example [10M DEC-17]