PROGRAM 8: GRAPH TRAVERSALS

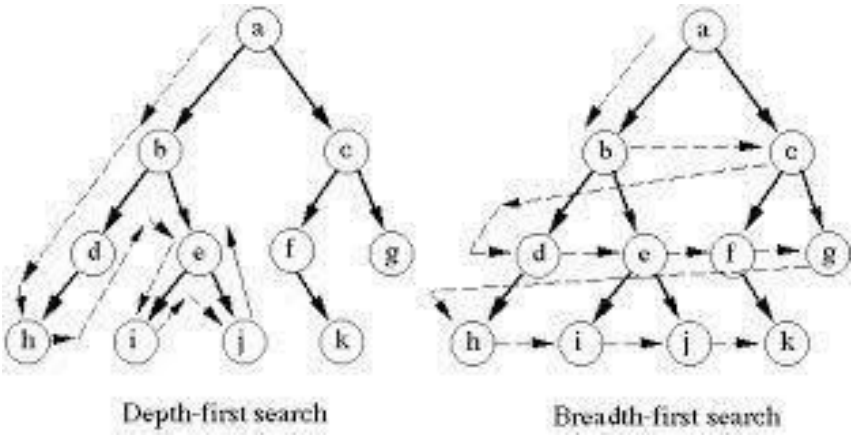
Aim: To write a program in C++ for implementing graph traversal.

Description:

Graph traversal is a technique used for searching a vertex in a graph. The graph traversal is also used to decide the order of vertices is visited in the search process. A graph traversal finds the edges to be used in the search process without creating loops. That means using graph traversal we visit all the vertices of the graph without getting into looping path.

There are two graph traversal techniques and they are as follows...

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



Algorithm:

Step 1: Insert the root node or starting node of a tree or a graph in the stack.

Step 2: Pop the top item from the stack and add it to the visited list.

Step 3: Find all the adjacent nodes of the node marked visited and add the ones that are not yet visited, to the stack.

Step 4: Repeat steps 2 and 3 until the stack is empty.

Program:

```
#include <iostream>
#include<vector>
int main()
{
cout << “DFS Traversal on a Graph \n";
//variable declaration
int cost[10][10], i, j, k, n, e, top, v, stk[10], visit[10], visited[10];
cout << "Enter the number of vertices in the Graph: ";
cin >> n;
cout << "\nEnter the number of edges in the Graph : ";
cin >> e;
cout << "\nEnter the start and end vertex of the edges: \n";

for (k = 1; k <= e; k++)
        {
            cin >> i >> j;
            cost[i][j] = 1;
        }
cout << "\nEnter the initial vertex to start the DFS traversal with: ";
cin >> v;
```

```
cout << "\nThe DFS traversal on the given graph is : \n";
cout << v << " ";
//As we start with the vertex v, marking it visited to avoid visiting again
visited[v] = 1;
k = 1;
//The DFS Traversal Logic
while (k < n)
{
        for (j = n; j >= 1; j--)
        {
           if (cost[v][j] != 0 && visited[j] != 1 && visit[j] != 1)
           {
               visit[j] = 1;
              //put all the vertices that are connected to the visited vertex
                into a stack
                stk[top] = j;
                top++;
            }
         }
//output all the connected vertices one at a time
        v = stk[--top];
        cout << v << " ";
        k++;
        //as v is visited so it is not a valid candidate to visit in future so
        visit[v]=0 and visited[v]=1
         visit[v] = 0;
```

```
        //to mark it visited
          visited[v] = 1;
}
        cout << "\n\n\n";
        return 0;
}
```

Output:

```
DFS Traversal on a Graph
Enter the number of vertices in the Graph: 4
Enter the number of edges in the Graph: 3
Enter the start and end vertex of the edges:
1
2
1
3
3
4
Enter the initial vertex to start the DFS traversal with: 1
The DFS traversal on the given graph is:
1 2 3 4
```

Result:

Thus a C++ program has been written and executed for implementing Graph Traversal.