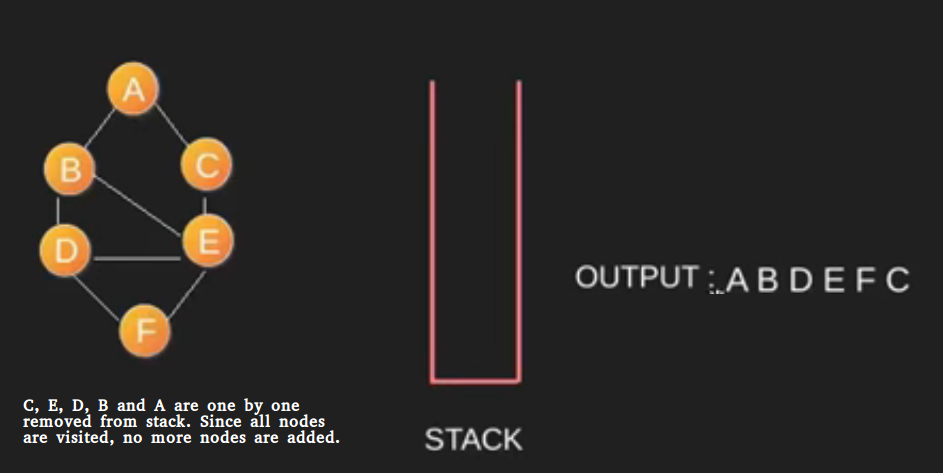
**Assignment 5**

**Aim :**

Represent a given graph using adjacency matrix and traverse each node using Depth first search.

**Theory:**

**Depth-first search** (**DFS**) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. It is similar to the depth first traversal of a tree, however, in order to ensure that roots are not repeated, we use a visited array. Depth first search can be performed recursively as well as iteratively. Stack is used as an assistive data structure in depth first search.



An example of depth first search.

**Algorithm**

DFS-iterative (G, s): //Where G is graph and s is source vertex

let S be stack

S.push( s ) //Inserting s in stack

mark s as visited.

while ( S is not empty):

//Pop a vertex from stack to visit next

v = S.top( )

S.pop( )

//Push all the neighbours of v in stack that are not visited

for all neighbours w of v in Graph G:

if w is not visited :

S.push( w )

mark w as visited

DFS-recursive(G, s):

mark s as visited

for all neighbours w of s in Graph G:

if w is not visited:

DFS-recursive(G, w)

**Program :**

**Problem Statement : Represvnt a given graph using adjacency matrix and traversv each node using Depth first svarch**

#include<iostream>

using namespace std;

void dfs\_rec(int v,int visit[],int u);

int adj[20][20];

int main()

{

int v,e,a,b;

cout<<"Enter the number of vertices :";

cin>>v;

// int adj[v][v];

for(int i=0;i<v;i++)

for(int j=0;j<v;j++)

{

adj[i][j] = 0;

}

cout<<"Enter the number of edges : ";

cin>>e;

for(int i=0;i<e;i++)

{

cout<<"Enter the edge (Starting and ending point) : ";

cin>>a>>b;

adj[a][b]= adj[b][a] = 1;

}

int visited[v];

for(int i=0;i<v;i++)

visited[i] =0;

cout<<"\t\tThe Depth First Search of your graph is : ";

dfs\_rec(0,visited,v);

cout<<"\n\n";

}

void dfs\_rec(int v,int visit[],int u)

{

cout<<v;

visit[v] = 1;

for(int j=0;j<u;j++)

{

if(adj[v][j]==1&&visit[j]==0)

{

dfs\_rec(j,visit,u);

}

}

}

/\*

//IF YOU WANT TO PRINT THE ADJANCENCY MATRIX OF GRAPH

cout<<"\n";

for(int i=0;i<v;i++)

{

for(int j=0;j<v;j++)

{

cout<<adj[i][j]<<"\t";

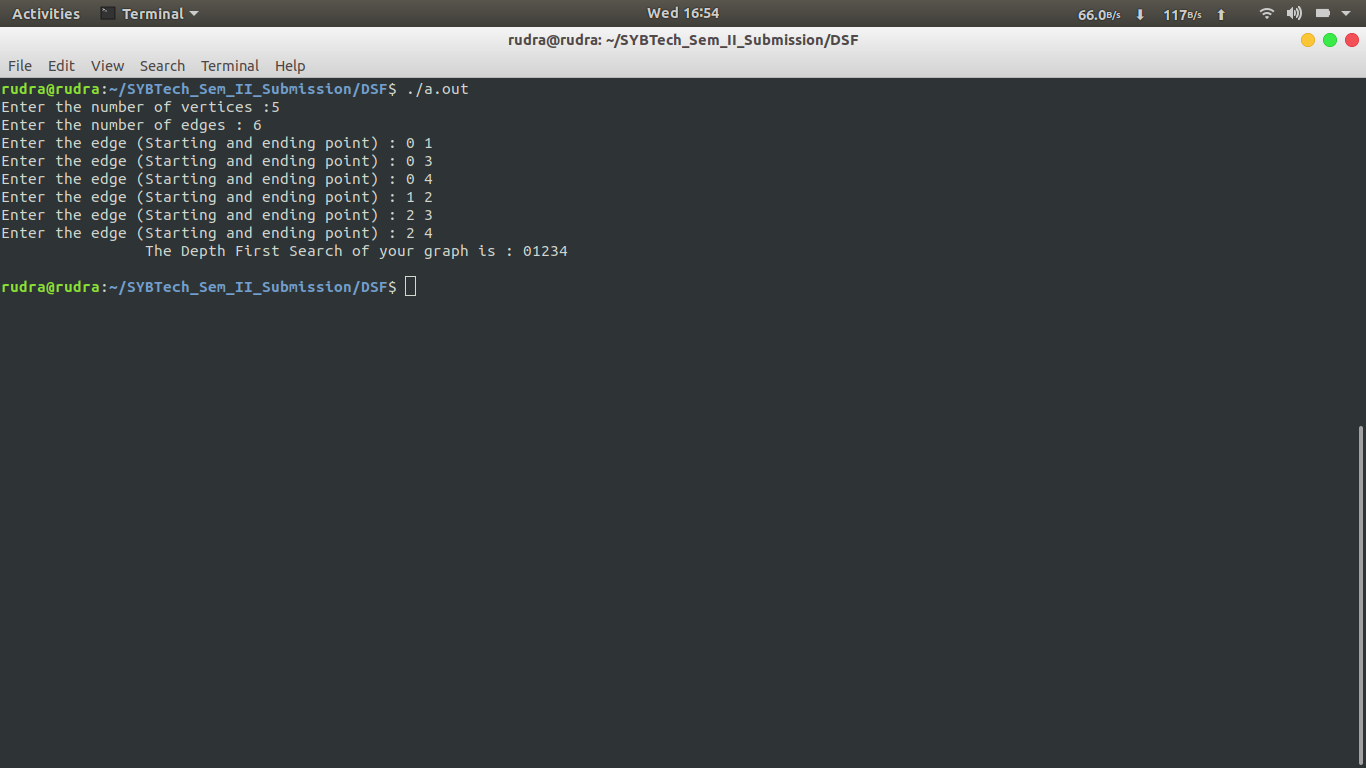
}

cout<<"\n";

}

\*/

**Output.**



**Conclusion :**

Through this assignment we learnt how to implement the Depth First Search