Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2021), B.Sc. in CSE (Day)

LAB REPORT NO 05

Course Code:206 Section:DB

Lab Experiment Name: DFS Traversal

Student Details

	Name	ID
1.	Shamim Ahmed	201902067

Submission Date : 27/07/2021

Course Teacher's Name : Monoshi Kumar Roy

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status Marks: ……

Signature:……

.

Comments:.....	**Date:.....**

1. TITLE OF THE LAB EXPERIMENT

(i) Traverse a graph using depth first search algorithm.
(ii) Using recursion to show to traversed vertices.
(iii) Using stack to traverse all vertices using DFS.
(iv) Which approach is more suitable in my opinion?

2. OBJECTIVES/AIM

From this lab we will learn how to traverse all vertices in a graph with using recursion and using stack with c++.

3. PROCEDURE / ANALYSIS / DESIGN

DFS: 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.

DFS 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.

Pseudocode:

```
Procedure DFS(G, x)
                G->graph to be traversed
                x->start node

 begin
         x.visited = true
        for each v belong to G.Adj[ x]
        if v.visited == false
                DFS{G,v)
 end procedure
int()
 {
         For each x belongto G
         x.visited = false
        For each x belong to G
                DFS(G, x)
 }
```

4. IMPLEMENTATION

Using recursion:

```
#include <bits/stdc++.h>
using namespace std;

vector <vector<int>>vec;
vector<bool> visited;


void dfs(int s){
    visited[s]=true;
    cout<<s<<" ";
    for(int i=0; i<vec[s].size();++i)
        {
            if(visited[vec[s][i]]==false)
            {
                dfs(vec[s][i]);
            }
        }

}
void initialization()
{
    for(int i=0; i<visited.size(); ++i)
        visited[i]=false;
}

int main()
{
   int node,edges,x,y;
   cin>>node>>edges;
   vec.resize(node+1);
   visited.resize(node+1);
   for(int i=0; i<edges;++i)
   {
    cin>>x>>y;
    vec[x].push_back(y);
```

```
    vec[y].push_back(x);
    }
    cout<<"Printing graph using DFS:"<<endl;
    initialization();
    dfs(1);
}


/* sir's graph

        1
       /  \
      2----5
     /      \
    3--------4
             \
              6

SO INPUT
6 7
1 5
1 2
2 5
2 3
3 4
4 6
4 5

*/
```

Using Stack :

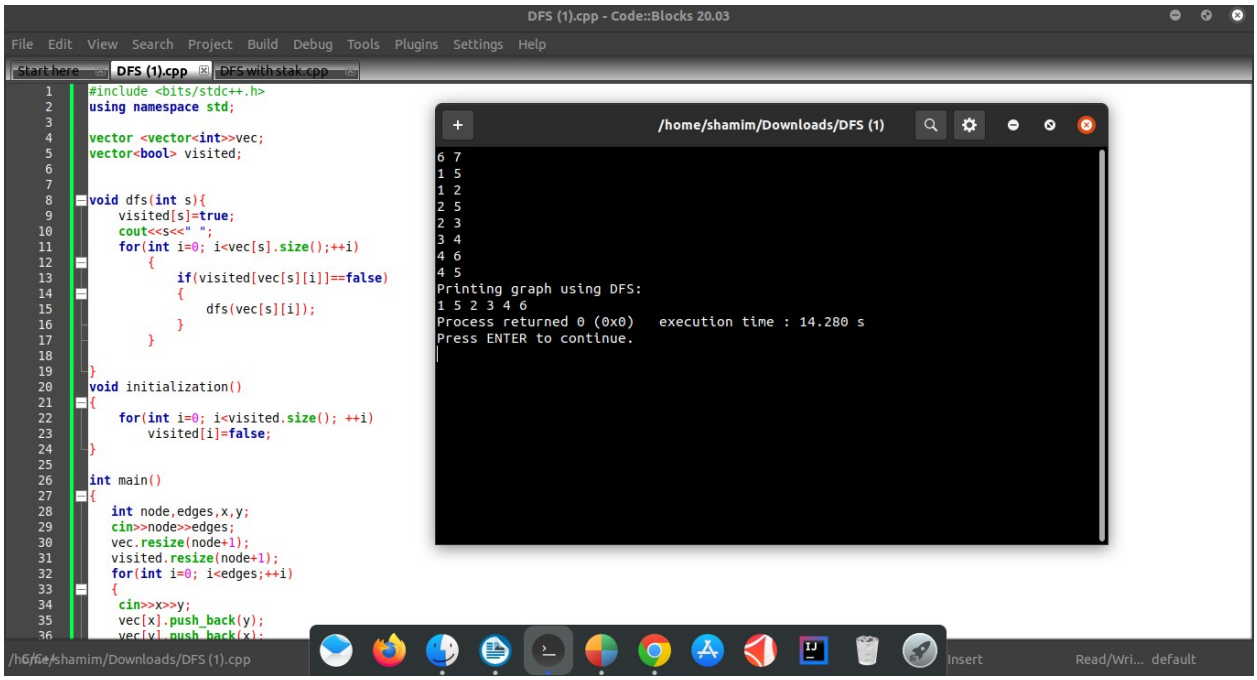
```
#include <bits/stdc++.h>
using namespace std;

vector <vector<int>>vec;
vector<bool> visited;
```

```
void dfs( const vector<vector<int>>& vec,int s){
   std::stack<int>stk;
   stk.push(s);
   visited[s]= true;
   while(!stk.empty())
   {
      int node= stk.top();
      stk.pop();
      cout<<node<<" ";
      for(int i=0; i<vec[node].size();++i)
      {
         if(!visited[vec[node][i]])
         {
            stk.push(vec[node][i]);
            visited[vec[node][i]]=true;
         }
      }
   }
}
void initialization()
{
   for(int i=0; i<visited.size(); ++i)
      visited[i]=false;
}

int main()
{
  int node,edges,x,y;
  cin>>node>>edges;
  vec.resize(node+1);
  visited.resize(node+1);
  for(int i=0; i<edges;++i)
  {
   cin>>x>>y;
   vec[x].push_back(y);
   vec[y].push_back(x);
   }
   cout<<"Printing graph using DFS:"<<endl;
```

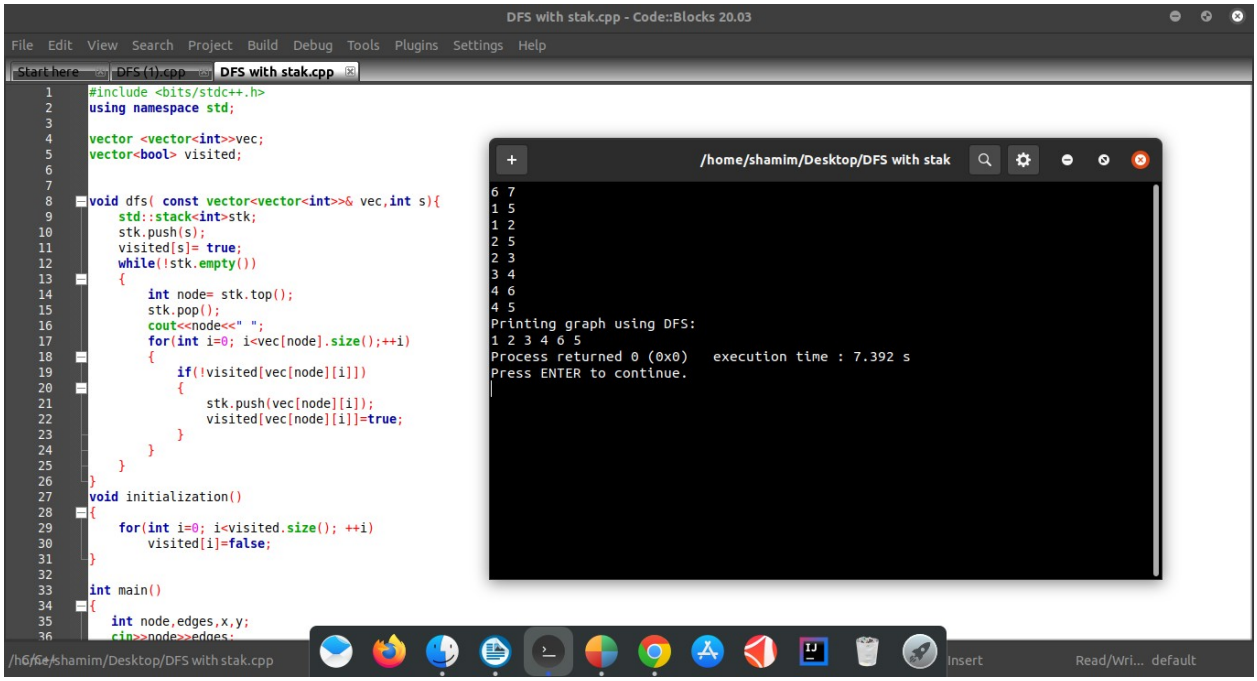
```
    initialization();
    dfs(vec,1);
}


/* sir's graph

        1
       /  \
      2----5
     /      \
    3--------4
             \
              6

SO INPUT
6 7
1 5
1 2
2 5
2 3
3 4
4 6
4 5

*/
```

5.TEST RESULT / OUTPUT

Using recursion:



Using Stack:



5. ANALYSIS AND DISCUSSION

In both using recursion and stack DFS traversal have same time complexity of O(V+E) where V is the number of vertices and E is the number of edges in the graph and Space Complexity is O(V). From my point of view recursion was much easier to implement. Using recursion was more suitable in my opinion.