## CHANDIGARH UNIVERSITY

## UNIVERSITY INSTITUTE OF NGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Submitted To:**  Vivek Kumar(21BCS8129) Neha Dutta(E12830) | |
| **Subject Name** | Design and Analysis of Algorithm Lab |
| **Subject Code** | 20CSP-312 |
| **Branch** | Computer Science and Engineering |
| **Semester** | 5th |

**Experiment - 8**

**Student Name: Vivek Kumar UID: 21BCS8129**

**Branch: BE-CSE(LEET) Section/Group: 20BCS-WM-616/A**

**Semester: 5th Date of Performance: 07/11/2022**

**Subject Name: DAA Lab Subject Code: 20CSP-312**

1. **Aim/Overview of the practical:**

Code and analyse to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze.

**2. Task to be done/ Which logistics used:**

Code and analyse to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze.

**3. Requirements (For programming-based labs):**

* Laptop or PC.
* Operation system (Mac, Windows, Linux, or any)
* Vs-Code with MinGw or any C++ Compiler

**4. Steps for experiment/practical/Code:**

#include <iostream>

#include <list>

using namespace std;

class DFSGraph

{

    int V;

    list<int> \*adjList;

    void DFS\_util(int v, bool visited[]);

public:

    DFSGraph(int V)

    {

        this->V = V;

        adjList = new list<int>[V];

    }

    void addEdge(int v, int w)

    {

        adjList[v].push\_back(w);

    }

    void DFS();

};

void DFSGraph::DFS\_util(int v, bool visited[])

{

    visited[v] = true;

    cout << v << " ";

    list<int>::iterator i;

    for (i = adjList[v].begin(); i != adjList[v].end(); ++i)

        if (!visited[\*i])

            DFS\_util(\*i, visited);

}

void DFSGraph::DFS()

{

    bool \*visited = new bool[V];

    for (int i = 0; i < V; i++)

        visited[i] = false;

        for (int i = 0; i < V; i++) if (visited[i] == false)

            DFS\_util(i, visited);

}

int main()

{

    int size,from,to;

    cout << "Enter the Number of Edge: " << endl;

    cin >> size;

    DFSGraph gdfs(size);

    while(true){

        cout << "Enter the From and To Edge respectively: " << endl;

        cin >> from >> to;

        gdfs.addEdge(from,to);

        if((size-1)== to && from==to)

            break;

    }

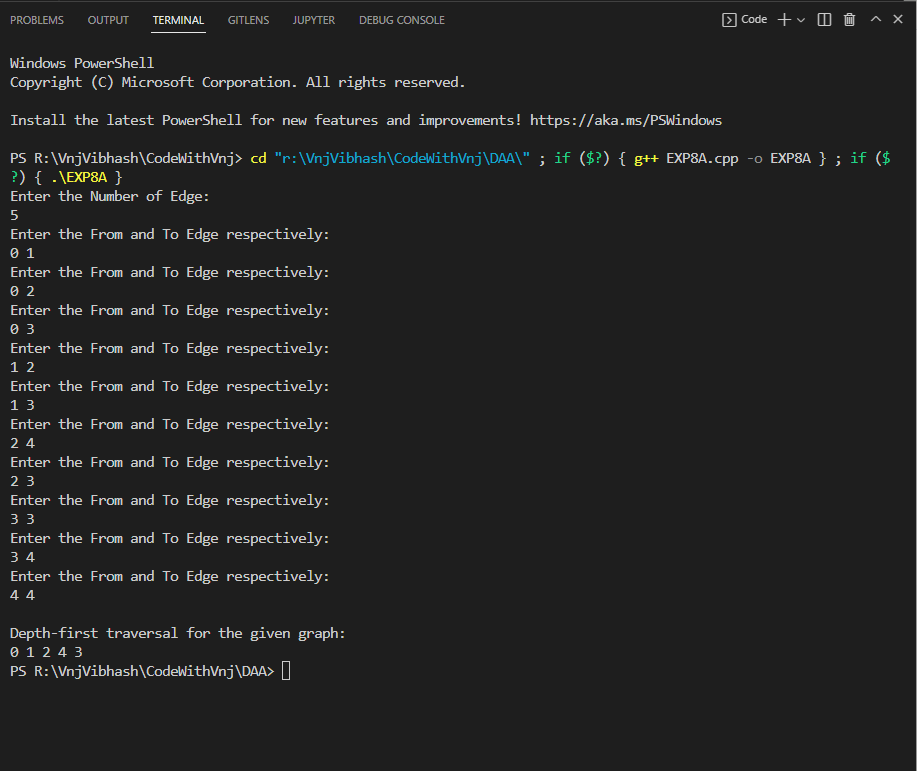
    cout << endl << "Depth-first traversal for the given graph:" << endl;

    gdfs.DFS();

    return 0;

}

**5. Output:**

****

**Learning outcomes (What I have learnt):**

1. How to solve the DFS using dynamic programming.

**Evaluation Grid (To be created per the faculty's SOP and Assessment guidelines):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Worksheet completion including writing learning objectives/Outcomes.  (To be submitted at the end of the day). |  |  |
| 2. | Post-Lab Quiz Result. |  |  |
| 3. | Student Engagement in  Simulation/Demonstration/Performance and Controls/Pre-Lab Questions. |  |  |
|  | Signature of Faculty (with Date): | Total Marks Obtained: |  |