मौलाना आज़ाद
राष्ट्रीय प्रौद्योगिकी संस्थान भोपाल (म.प्र.) भारत
MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL (M. P.) INDIA

ADA Lab

Assignment - 8

Dated: 31-03-2021

Sub Code: CSE-228

Vivek Kumar Ahirwar
191112419
CSE - 3

Department of
Computer Science and Engineering

Subject Coordinator:
Prof. Manish Pandey

Contents

Problem 1: Graph BFS DFS

Design, Develop and Implement a Program in your preferred language for the following operations on Graph(G) of Cities (Take graph of your choice)

a) Create a Graph of N cities using Adjacency Matrix.

b) Print all the nodes reachable from a given starting node in a digraph using BFS method.

c) Check whether a given graph is connected or not using DFS method

Code

```
// Keep Changing....@Vi

#include <iostream>
using namespace std;

class Graph
{

public:
    bool **adjMatrix;
    int numVertices;
    // Initialize the matrix to zero
    Graph(int numVertices)
    {
        this->numVertices = numVertices;
        adjMatrix = new bool *[numVertices];
        for (int i = 0; i < numVertices; i++)
        {
            adjMatrix[i] = new bool[numVertices];
            for (int j = 0; j < numVertices; j++)
                adjMatrix[i][j] = false;
        }
    }

    void addEdge(int i, int j)
    {
        adjMatrix[i][j] = true;
        adjMatrix[j][i] = true;
    }

    void bfs(int vertex)
    {
        int reach[numVertices];
        bool vis[numVertices];
        for (int i = 0; i < numVertices; i++)
        {
            reach[i] = -1;
            vis[i] = false;
        }
        int start = 0, end = 0;
        reach[start] = vertex;
        vis[vertex] = true;
        while (start <= end)
        {
            int u = reach[start];
            start++;
            for (int i = 0; i < numVertices; i++)
            {
                if (vis[i] == false && adjMatrix[u][i] == true)
                {
                    reach[++end] = i;
                    vis[i] = true;
                }
            }
        }
        cout << "vertex reachable from vertex " << vertex << " are : ";
        for (int i = 1; i <= end; i++)
        {
            cout << reach[i] << " ";
        }
    }

    void dfs(int vertex, bool vis[], int &cnt)
    {
        vis[vertex] = true;
        cnt++;
        for (int i = 0; i < numVertices; i++)
        {
            if (vis[i] == false && adjMatrix[vertex][i] == true)
            {
                dfs(i, vis, cnt);
            }
        }
    }

    void connected()
    {
        bool vis[numVertices];
        for (int i = 0; i < numVertices; i++)
        {
            vis[i] = false;
        }
        int cnt = 0;
```

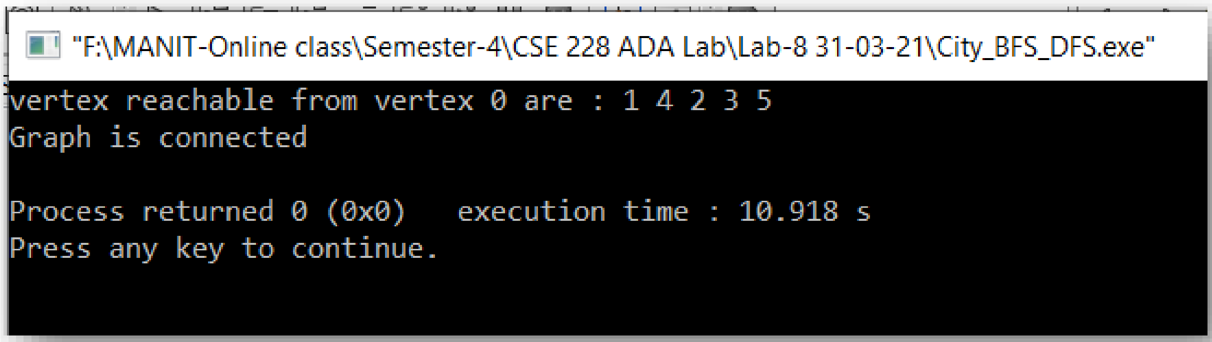
```
        dfs(0, vis, cnt);
        if (cnt == numVertices)
        {
            cout << "\nGraph is connected\n";
        }
        else
        {
            cout << "\nGraph is not connected\n";
        }
    }
};

int main()
{
    Graph g(6);
    g.addEdge(0, 1);
    g.addEdge(1, 2);
    g.addEdge(4, 0);
    g.addEdge(4, 1);
    g.addEdge(2, 3);
    g.addEdge(3, 1);
    g.addEdge(3, 5);

    g.bfs(0);

    g.connected();
}
```

Output



```
"F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-8 31-03-21\City_BFS_DFS.exe"
vertex reachable from vertex 0 are : 1 4 2 3 5
Graph is connected

Process returned 0 (0x0)   execution time : 10.918 s
Press any key to continue.
```

Analysis

Time Complexity: O(N).
For BFS/DFS

Auxiliary Space: O(N*N).
For storing adjacency matrix