G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Assignment no :- 9

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	13-12-2021
Subject Name /Code	Data Structures and Algorithms/ UCSL201/UCSP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Experiment No 9

Aim: Implement graph using adjacency list or matrix and perform DFS or BFS.

Theory:

- **ALGORITHMS:**

1. Declare an array of pointers to a link list having a data field (to store vertex numbers) and a forward pointer. The number of array of pointers would equal the total number of vertices in the graph.
2. Take the edge set from the user. If for eg, vertex 1 is connected to vertex 2 and 3 in the graph, the 1st location of the array of pointers (corresponding to vertex 1) would point to 2 nodes, one having the data 2 (corresponding to vertex 2) and the other having data 3
3. In this way construct the entire adjacency list.

- **DFS (Depth First Search)**

1. The start vertex is visited. Next an unvisited vertex W adjacent to v is selected and a DFS from W initiated.
2. When a vertex u is reached such that all its adjacent vertices have been visited. We back up to the last vertex visited which has an unvisited vertex W adjacent to it and initiate a DFS search from W.
3. The search terminates when no unvisited vertex can be reached from any of the visited ones.

BFS (Breadth First Search)

1. Starting at vertex v and marking it as visited. BFS differs from DFS in that all unvisited vertices adjacent to v are visited next.

2. Then unvisited vertices adjacent to these vertices are visited and so on.

3. A queue is used to store vertices as they are visited so that later search can be initiated from those vertices.

Test CONDITIONS:

Enter the graph with 8 vertices and 10 edges (1,2), (1,3), (2,4), (2,5), (3,6), (3,7) (4,8), (5,8), (6,8), (7,8)

The order of the vertices visited by DFS is: 1,2,4,8,5,6,3,7
The order of the vertices visited by BFS is: 1,2,3,4,5,6,7,8.

INPUT:→
The number of vertices and the edge set of the graph.

OUTPUT:→
The order of vertices visited in both DFS and BFS.

Program code :-

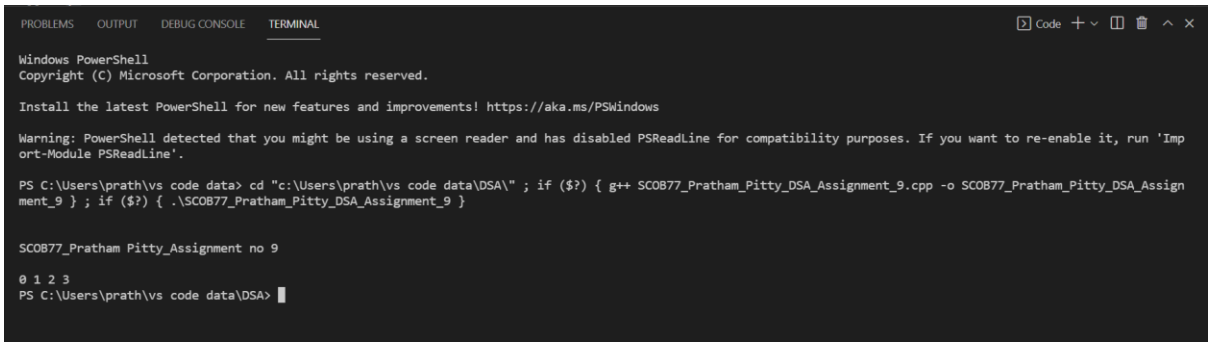
```
// C++ implementation of the approach
#include <bits/stdc++.h>
using namespace std;

class Graph
{

    // Number of vertex
    int v;

    // Number of edges
    int e;

    // Adjacency matrix
    int **adj;

public:
    // To create the initial adjacency matrix
    Graph(int v, int e);

    // Function to insert a new edge
    void addEdge(int start, int e);

    // Function to display the BFS traversal
    void BFS(int start);
};

// Function to fill the empty adjacency matrix
```

```
Graph::Graph(int v, int e)
{
    this->v = v;
    this->e = e;
    adj = new int *[v];
    for (int row = 0; row < v; row++)
    {
        adj[row] = new int[v];
        for (int column = 0; column < v; column++)
        {
            adj[row][column] = 0;
        }
    }
}

// Function to add an edge to the graph
void Graph::addEdge(int start, int e)
{

    // Considering a bidirectional edge
    adj[start][e] = 1;
    adj[e][start] = 1;
}

// Function to perform BFS on the graph
void Graph::BFS(int start)
{
    // Visited vector to so that
    // a vertex is not visited more than once
```

```
// Initializing the vector to false as no
// vertex is visited at the beginning
vector<bool> visited(v, false);
vector<int> q;
q.push_back(start);

// Set source as visited
visited[start] = true;

int vis;
while (!q.empty())
{
    vis = q[0];

    // Print the current node
    cout << vis << " ";
    q.erase(q.begin());

    // For every adjacent vertex to the current vertex
    for (int i = 0; i < v; i++)
    {
        if (adj[vis][i] == 1 && (!visited[i]))
        {

            // Push the adjacent node to the queue
            q.push_back(i);

            // Set
            visited[i] = true;
```

```
        }
      }
    }
}

// Driver code
int main()
{
    cout << "\n\nSCOB77_Pratham Pitty_Assignment no 9 \n\n";

    int v = 5, e = 4;

    // Create the graph
    Graph G(v, e);
    G.addEdge(0, 1);
    G.addEdge(0, 2);
    G.addEdge(1, 3);

    G.BFS(0);
}
```

Output :-



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                    Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Imp
ort-Module PSReadLine'.

PS C:\Users\prath\vs code data> cd "c:\Users\prath\vs code data\DSA\" ; if ($?) { g++ SCOB77_Pratham_Pitty_DSA_Assignment_9.cpp -o SCOB77_Pratham_Pitty_DSA_Assign
ment_9 } ; if ($?) { .\SCOB77_Pratham_Pitty_DSA_Assignment_9 }


SCOB77_Pratham Pitty_Assignment no 9

0 1 2 3
PS C:\Users\prath\vs code data\DSA>
```