Assignment No:9

Aim:

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

Theory:

Algorithms ⇒

Creation of Adjacency list ⇒

Declare array of pointers to a linklist having a data field and forward pointers. The no.of array of pointers would point to 2 nodes one having the data 2 and other having data 3.

In this way construct the entire adjacency list.

Depth First Search ⇒

The start vertex is visited. Next an unvisited vertex w adjacent to v is selected and DFS from w initialated.

When a vertex u is reached such that all its adjocents vertices have been visited. We both as unvisited vertex w adjacent to it and initiatie a DFS from w.

The search terminates when no unvisited vertex can be reached from any of the visited once.

Breadth First Search ⇒

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

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

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

Test Condition ⇒

Graph of 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 vertices visited by DFS is,

1, 2, 3, 4, 8, 5, 6, 7

The order of vertices visited by BFS is,

1, 2, 3, 4, 5, 6, 7, 8

Input ⇒

The no of vertices and the edge set of graph.

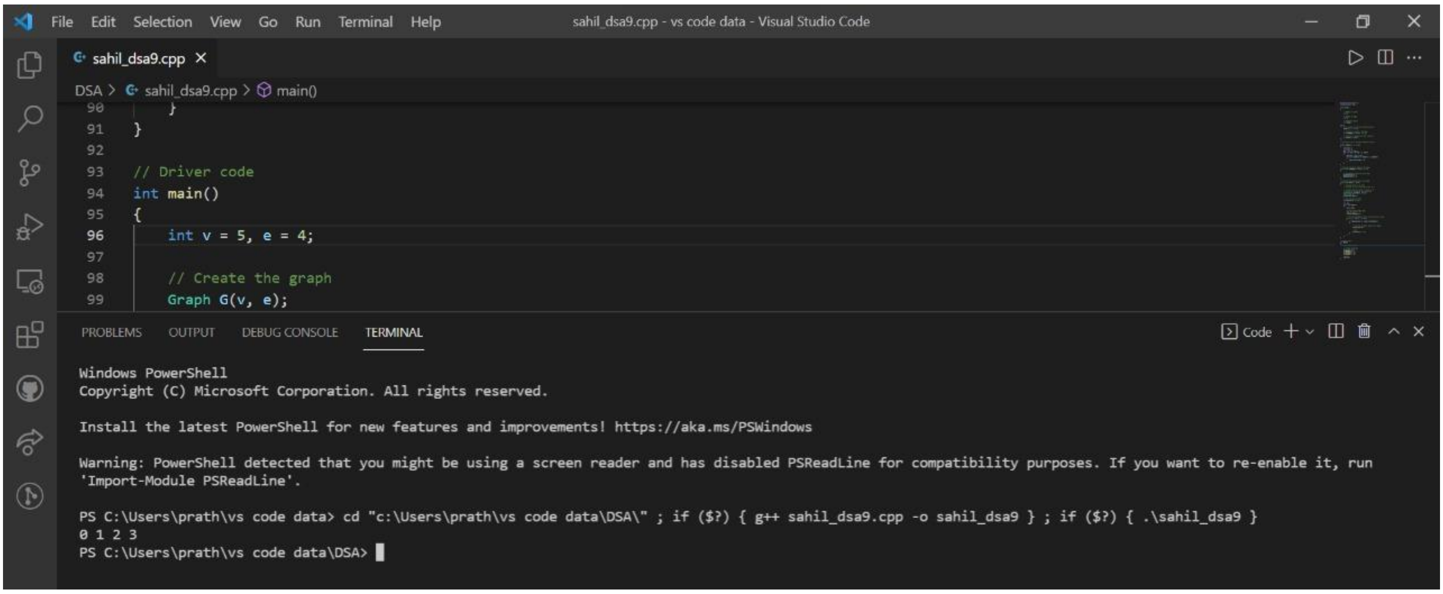
Output ⇒

The order of vertices visited in both DFS & BFS.

Program Code:-

```
#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() {
  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);
}
```

Program Output :



File Edit Selection View Go Run Terminal Help
sahil_dsa9.cpp - vs code data - Visual Studio Code
sahil_dsa9.cpp
DSA > sahil_dsa9.cpp > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code

```cpp
    }
}

// Driver code
int main()
{
    int v = 5, e = 4;

    // Create the graph
    Graph G(v, e);
```

```
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
'Import-Module PSReadLine'.

PS C:\Users\prath\vs code data> cd "c:\Users\prath\vs code data\DSA\" ; if ($?) { g++ sahil_dsa9.cpp -o sahil_dsa9 } ; if ($?) { .\sahil_dsa9 }
0 1 2 3
PS C:\Users\prath\vs code data\DSA> █
```