BRACT’s

Vishwakarma Institute of Information Technology, Pune

**Practical Implementation Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **Department:** IT | **Semester:** IV | **Academic Year:** 2024-25 | **Practical No: 2** |
| **Class/ Division/ Batch: SY (B)- B3** | | **Roll no: 70** | |
| **Course:** Data Structures and Analysis of Algorithms | | **Name of Student**:Anushka Kadam | |

**Aim:** Create a graph and perform graph traversal using BFS and DFS.

**Code:**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

class Graph{

int V;

vector<vector<int>> adjList;

public:

Graph(int V)

{

this->V = V;

adjList.resize(V);

}

void addEdge(int v, int w)

{

adjList[v].push\_back(w);

adjList[w].push\_back(v);

}

void BFS(int start)

{

vector<bool> visited(V, false);

queue<int> q;

visited[start] = true;

q.push(start);

cout << "BFS Traversal starting from vertex " << start << ": ";

while (!q.empty())

{

int node = q.front();

cout << node << " ";

q.pop();

for (auto neighbor : adjList[node])

{

if (!visited[neighbor])

{

visited[neighbor] = true;

q.push(neighbor);

}

}

}

cout << endl;

}

void DFSUtil(int node, vector<bool>& visited) {

visited[node] = true;

cout << node << " ";

for (auto neighbor : adjList[node])

{

if (!visited[neighbor]) {

DFSUtil(neighbor, visited);

}

}

}

void DFS(int start){

vector<bool> visited(V, false);

cout << "DFS Traversal starting from vertex " << start << ": ";

DFSUtil(start, visited);

cout << endl;

}

};

int main()

{

int V = 6;

Graph g(V);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 3);

g.addEdge(1, 4);

g.addEdge(2, 4);

g.addEdge(3, 5);

g.addEdge(4, 5);

g.BFS(0);

g.DFS(0);

return 0;

}

int main() {

int V, E;

cout << "Enter number of vertices: ";

cin >> V;

Graph g(V);

cout << "Enter number of edges: ";

cin >> E;

cout << "Enter " << E << " edges (as pairs of vertex indices like 0 1):" << endl;

for (int i = 0; i < E; ++i) {

int u, v;

cin >> u >> v;

g.addEdge(u, v);

}

int start;

cout << "Enter starting vertex for BFS and DFS: ";

cin >> start;

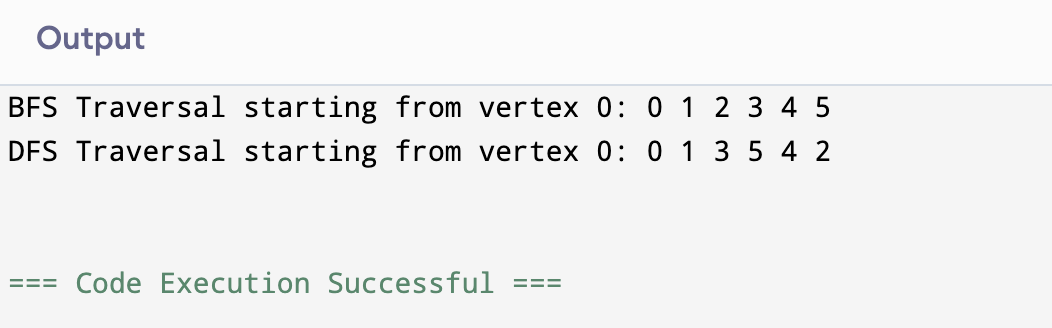
g.BFS(start);

g.DFS(start);

return 0;

}

**Output:**

****