**ASSIGNMENT 2**

Name: Swaraj Murlidhar Patil

Div : CSAI B Roll No: 15

PRN: 12310234

**Implement Uninformed Search.**

**1)BREADTH FIRST SEARCH (BFS)**

**CODE:**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void bfs(const vector<vector<int>>& graph, int start) {

    vector<bool> visited(graph.size(), false);

    queue<int> q;

    visited[start] = true;

    q.push(start);

    while (!q.empty()) {

        int node = q.front();

        q.pop();

        cout << node << " ";

        for (int neighbor : graph[node]) {

            if (!visited[neighbor]) {

                visited[neighbor] = true;

                q.push(neighbor);

            }

        }

    }

}

int main() {

    int n = 5;

    vector<vector<int>> graph(n);

    graph[0].push\_back(1);

    graph[1].push\_back(0);

    graph[1].push\_back(3);

    graph[3].push\_back(1);

    graph[0].push\_back(2);

    graph[2].push\_back(0);

    graph[2].push\_back(3);

    graph[3].push\_back(2);

    graph[3].push\_back(4);

    graph[4].push\_back(3);

    cout << "BFS starting from node 0: ";

    bfs(graph, 0);

    cout << endl;

    return 0;

}

**OUTPUT:**

BFS starting from node 0: 0 1 2 3 4

**2)DEPTH FIRST SEARCH (DFS)**

**CODE:**

#include <iostream>

#include <vector>

using namespace std;

void DFS(int node, const vector<vector<int>>& adjacencyList, vector<bool>& visited) {

    visited[node] = true;

    cout << "Visited Node: " << node << endl;

    for (int neighbor : adjacencyList[node]) {

        if (!visited[neighbor]) {

            DFS(neighbor, adjacencyList, visited);

        }

    }

}

int main() {

    int numNodes = 6;

    vector<vector<int>> adjacencyList(numNodes);

    adjacencyList[0] = {1, 2};

    adjacencyList[1] = {0, 3, 4};

    adjacencyList[2] = {0, 4};

    adjacencyList[3] = {1, 5};

    adjacencyList[4] = {1, 2, 5};

    adjacencyList[5] = {3, 4};

    vector<bool> visited(numNodes, false);

    DFS(0, adjacencyList, visited);

return 0;

}

**OUTPUT:**

Visited Node: 0

Visited Node: 1

Visited Node: 3

Visited Node: 5

Visited Node: 4

Visited Node: 2

**3)DEPTH FIRST LIMIT SEARECH**

**CODE:**

#include <iostream>

#include <vector>

using namespace std;

void DFLS(int node, const vector<vector<int>>& adjacencyList, vector<bool>& visited, int depth, int limit) {

// If the depth limit is reached, stop exploring

if (depth > limit) {

return;

}

visited[node] = true;

cout << "Visited Node: " << node << " at Depth: " << depth << endl;

for (int neighbor : adjacencyList[node]) {

if (!visited[neighbor]) {

DFLS(neighbor, adjacencyList, visited, depth + 1, limit);

}

}

}

int main() {

int numNodes = 6;

vector<vector<int>> adjacencyList(numNodes);

adjacencyList[0] = {1, 2};

adjacencyList[1] = {0, 3, 4};

adjacencyList[2] = {0, 4};

adjacencyList[3] = {1, 5};

adjacencyList[4] = {1, 2, 5};

adjacencyList[5] = {3, 4};

int limit = 2; // Depth limit for DFLS

vector<bool> visited(numNodes, false);

DFLS(0, adjacencyList, visited, 0, limit); // Starting from node 0 with initial depth 0

return 0;

}

**OUTPUT:**

Visited Node: 0 at Depth: 0

Visited Node: 1 at Depth: 1

Visited Node: 3 at Depth: 2

Visited Node: 4 at Depth: 2

Visited Node: 2 at Depth: 1