Aim-dfs

#include <iostream>

#include <unordered\_map>

#include <vector>

#include <stack>

#include <set>

using namespace std;

class Graph {

public:

unordered\_map<int, vector<int>> adjList;

void addVertex(int vertex) {

adjList[vertex]; // Automatically creates an empty list for the vertex if it doesn't exist

}

void addEdge(int vertex1, int vertex2) {

adjList[vertex1].push\_back(vertex2); // Directed edge from vertex1 to vertex2

}

void dfs(int start) {

set<int> visited; // To track visited nodes

stack<int> s;

// Start DFS from the 'start' vertex

s.push(start);

while (!s.empty()) {

int vertex = s.top();

s.pop();

// If vertex hasn't been visited yet, process it

if (visited.find(vertex) == visited.end()) {

visited.insert(vertex);

cout << vertex << " "; // Visit the current node (print it)

// Add all unvisited adjacent vertices to the stack

for (int neighbor : adjList[vertex]) {

if (visited.find(neighbor) == visited.end()) {

s.push(neighbor);

}

}

}

}

}

};

int main() {

Graph g;

g.addVertex(1);

g.addVertex(2);

g.addVertex(3);

g.addVertex(4);

// Add edges (directed graph)

g.addEdge(1, 2);

g.addEdge(1, 3);

g.addEdge(2, 4);

cout << "DFS Traversal starting from vertex 1: ";

g.dfs(1); // Perform DFS starting from vertex 1

cout << endl;

return 0;

}