//C++ Program: Telecommunication Network Optimization

#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

// Structure to represent a weighted edge in the graph

struct Edge {

int to, cost;

};

// Function to find the shortest path using Dijkstra's algorithm

vector<int> dijkstra(int source, int nodes, vector<vector<Edge>>& graph) {

vector<int> dist(nodes, INT\_MAX); // Distance from source to each node

dist[source] = 0;

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<>> pq; // Min-heap

pq.push({0, source});

while (!pq.empty()) {

int currentDist = pq.top().first;

int currentNode = pq.top().second;

pq.pop();

// Skip processing if the current distance is outdated

if (currentDist > dist[currentNode]) continue;

for (const auto& edge : graph[currentNode]) {

int nextNode = edge.to;

int weight = edge.cost;

if (dist[currentNode] + weight < dist[nextNode]) {

dist[nextNode] = dist[currentNode] + weight;

pq.push({dist[nextNode], nextNode});

}

}

}

return dist;

}

int main() {

int nodes, edges;

cout << "Enter the number of nodes and edges in the network: ";

cin >> nodes >> edges;

vector<vector<Edge>> graph(nodes);

cout << "Enter the edges in the format: from to cost" << endl;

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

int from, to, cost;

cin >> from >> to >> cost;

graph[from].push\_back({to, cost});

graph[to].push\_back({from, cost}); // Uncomment this line if the graph is undirected

}

int source;

cout << "Enter the source node: ";

cin >> source;

// Compute the shortest paths from the source node

vector<int> shortestPaths = dijkstra(source, nodes, graph);

cout << "Shortest path from node " << source << " to all other nodes:" << endl;

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

if (shortestPaths[i] == INT\_MAX) {

cout << "Node " << i << ": No path" << endl;

} else {

cout << "Node " << i << ": " << shortestPaths[i] << endl;

}

}

return 0;

}