

**Assignment #1**



**Date of Performance: 22-04-2024 Student ID: 20220104147**

**Date of Submission: 29-04-2024 Name: Md. Redwan Hossen**

**Group: C2**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0 based \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//**

**#include<bits/stdc++.h>**

**using namespace std;**

**#define WHITE 1**

**#define GRAY 2**

**#define BLACK 3**

**#define INF 1e9**

**#define NIL -1**

**int adj[100][100];**

**int color[100];**

**int node, edge;**

**vector<int> vktr;**

**vector<int> pre\_node[100];**

**int dist[100];**

**void Initialize\_Single\_Source(int source)**

**{**

**for(int i = 0; i < node; i++)**

**{**

**dist[i] = INF;**

**pre\_node[i].clear();**

**//pre\_node[i].push\_back(NIL);**

**}**

**dist[source] = 0;**

**}**

**void Relax(int u, int v, int w)**

**{**

**if(dist[v] > dist[u] + w)**

**{**

**dist[v] = dist[u] + w;**

**pre\_node[v] = pre\_node[u];**

**pre\_node[v].push\_back(u);**

**}**

**}**

**void dfsVisit(int x)**

**{**

**color[x] = GRAY;**

**//cout << x << " ";**

**for(int i = 0; i < node; i++)**

**{**

**if(adj[x][i] != 0 && color[i] == WHITE)**

**{**

**dfsVisit(i);**

**}**

**}**

**color[x] = BLACK;**

**vktr.push\_back(x);**

**//cout << x << " "; // Ulta print korte hbe.**

**}**

**void dfs()**

**{**

**for(int i = 0; i < node; i++)**

**{**

**color[i] = WHITE;**

**}**

**for(int i = 0; i < node; i++)**

**{**

**if(color[i] == WHITE)**

**{**

**dfsVisit(i);**

**}**

**}**

**}**

**int main()**

**{**

**//freopen("offline1.txt","r", stdin);**

**cout << "Enter the number of nodes & edges : ";**

**cin >> node >> edge ;**

**int n1, n2, n3;**

**cout << "Enter edges with weights:" << endl;**

**for(int i = 0; i < edge; i++)**

**{**

**cin >> n1 >> n2 >> n3;**

**//adj[n1][n2] = 1;**

**//adj[n2][n1] = 1;**

**adj[n1][n2] = n3;**

**}**

**dfs();**

**cout << "Topological Sort : ";**

**for(int i = vktr.size() - 1; i >= 0; i--)**

**{**

**cout << vktr[i] << " ";**

**}**

**cout << endl;**

**int source;**

**cout << "Enter the source node : ";**

**cin >> source;**

**Initialize\_Single\_Source(source);**

**for(int i = vktr.size() - 1; i >= 0; i--)**

**{**

**for(int j = 0; j < node; j++)**

**{**

**if(adj[vktr[i]][j] != 0)**

**{**

**Relax(vktr[i], j, adj[vktr[i]][j]);**

**}**

**}**

**}**

**cout << "Shortest paths from node " << source << endl;**

**for(int i = 0; i < node; i++)**

**{**

**cout << "Node " << i << ": ";**

**if(dist[i] == INF)**

**cout << "No path";**

**else**

**{**

**cout << "Cost: " << dist[i] << ", Path: ";**

**for(int j = 0; j < pre\_node[i].size(); j++)**

**{**

**cout << pre\_node[i][j] << " -> ";**

**}**

**cout << i;**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**/\***

**INPUT:**

**6 10**

**0 1 5**

**0 2 3**

**1 2 2**

**1 3 6**

**2 3 7**

**2 4 4**

**2 5 2**

**3 4 -1**

**3 5 1**

**4 5 -2**

**1**

**OUTPUT:**

**Enter the number of nodes & edges : 6 10**

**Enter edges with weights:**

**0 1 5**

**0 2 3**

**1 2 2**

**1 3 6**

**2 3 7**

**2 4 4**

**2 5 2**

**3 4 -1**

**3 5 1**

**4 5 -2**

**Topological Sort : 0 1 2 3 4 5**

**Enter the source node : 1**

**Shortest paths from node 1**

**Node 0: No path**

**Node 1: Cost: 0, Path: 1**

**Node 2: Cost: 2, Path: 1 -> 2**

**Node 3: Cost: 6, Path: 1 -> 3**

**Node 4: Cost: 5, Path: 1 -> 3 -> 4**

**Node 5: Cost: 3, Path: 1 -> 3 -> 4 -> 5**

**\*/**