Name: Rohan Arun Nalawade

Roll No: 231012

PRN: 22310407

SY IT A

Assignment 5(A)

Aim:

A business house has several offices in different countries; they want to lease phone n lines to connect them with each other and the phone company charges different rent to connect different pairs of cities. Business house wants to connect all its offices with a minimum total cost. Represent using appropriate data structure.

Code:

#include <iostream>

#include <climits>

#include <algorithm>

using namespace std;

struct Edge {

int u, v, cost;

};

class BusinessNetwork {

private:

int offices, edges;

int adjMatrix[100][100]; // Maximum 100 offices

Edge edgeList[1000]; // Maximum 1000 edges

public:

BusinessNetwork(int n) {

offices = n;

edges = 0;

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

for (int j = 0; j < offices; j++) {

adjMatrix[i][j] = 0;

}

}

}

void inputAdjacencyMatrix() {

cout << "Enter adjacency matrix (0 if no direct connection exists):\n";

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

for (int j = 0; j < offices; j++) {

cin >> adjMatrix[i][j];

if (adjMatrix[i][j] != 0 && i < j) { // Avoid duplicate edges

edgeList[edges].u = i;

edgeList[edges].v = j;

edgeList[edges].cost = adjMatrix[i][j];

edges++;

}

}

}

}

void sortEdges() {

sort(edgeList, edgeList + edges, [](Edge a, Edge b) {

return a.cost < b.cost;

});

}

int findParent(int node, int parent[]) {

if (parent[node] == node)

return node;

return parent[node] = findParent(parent[node], parent);

}

void unionSets(int u, int v, int parent[], int rank[]) {

int rootU = findParent(u, parent);

int rootV = findParent(v, parent);

if (rootU != rootV) {

if (rank[rootU] > rank[rootV])

parent[rootV] = rootU;

else if (rank[rootU] < rank[rootV])

parent[rootU] = rootV;

else {

parent[rootV] = rootU;

rank[rootU]++;

}

}

}

void kruskalMST() {

sortEdges();

int parent[100], rank[100];

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

parent[i] = i;

rank[i] = 0;

}

int totalCost = 0, count = 0;

cout << "Minimum Cost Phone Line Connections:\n";

for (int i = 0; i < edges && count < offices - 1; i++) {

int u = edgeList[i].u, v = edgeList[i].v, cost = edgeList[i].cost;

if (findParent(u, parent) != findParent(v, parent)) {

cout << "Office " << u << " - Office " << v << " : Cost " << cost << "\n";

totalCost += cost;

unionSets(u, v, parent, rank);

count++;

}

}

cout << "Minimum Total Cost: " << totalCost << endl;

}

};

int main() {

int offices;

cout << "Enter the number of offices: ";

cin >> offices;

BusinessNetwork network(offices);

network.inputAdjacencyMatrix();

network.kruskalMST();

return 0;

}

Output:

