Problem Statement:

You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

#include <iostream>

#include <vector>

#include <cstdint>

#define MAX\_NUM\_CITIES 10

using namespace std;

struct edge {

int start;

int end;

int wt;

};

class graph {

int adj\_mat[MAX\_NUM\_CITIES][MAX\_NUM\_CITIES] = {0};

string city\_names[MAX\_NUM\_CITIES];

int city\_count;

edge mst[MAX\_NUM\_CITIES - 1];

void add\_to\_list(vector<edge> &, edge);

int cost;

public:

graph();

void prims\_algo(int);

void display\_mst();

};

void graph::add\_to\_list(vector<edge> &list, edge e) {

list.push\_back(e);

for (int i = list.size() - 1; i > 0; i--) {

if (list[i].wt < list[i - 1].wt) {

swap(list[i], list[i - 1]);

} else {

break;

}

}

}

graph::graph() {

cost = 0;

cout << "Number of cities are (1-" << MAX\_NUM\_CITIES << "):\t";

cin >> city\_count;

city\_count = (city\_count > MAX\_NUM\_CITIES) ? MAX\_NUM\_CITIES : city\_count;

for (int i = 0; i < city\_count; i++) {

cout << "Enter city:\n" << i + 1 << ":\t";

cin >> city\_names[i];

}

// initialize all matrix with max value

for (int i = 0; i < city\_count; i++)

for (int j = 0; j < city\_count; j++) adj\_mat[i][j] = INT32\_MAX;

int num\_pairs;

cout << "Number of city pairs are:\t";

cin >> num\_pairs;

cout << "City codes are:\t" << endl;

for (int i = 0; i < city\_count; i++) {

cout << i << " - " << city\_names[i] << endl;

}

int x, y, wt;

for (int i = 0; i < num\_pairs; i++) {

cout << "Enter pair:\n" << i + 1 << ":\t";

cin >> x >> y;

cout << "Enter cost between city " << city\_names[x] << " & city "

<< city\_names[y] << ":\t";

cin >> wt;

adj\_mat[x][y] = wt;

adj\_mat[y][x] = wt;

}

}

void graph::prims\_algo(int start) {

bool visited[MAX\_NUM\_CITIES] = {0};

int visited\_count = 1;

visited[start] = 1;

vector<edge> adj;

for (int i = 0; i < city\_count; i++) {

if (adj\_mat[start][i] != INT32\_MAX) {

edge e;

e.start = start;

e.end = i;

e.wt = adj\_mat[start][i];

add\_to\_list(adj, e);

}

}

while (visited\_count != city\_count) {

edge m = adj.front();

adj.erase(adj.begin());

if (!visited[m.end]) {

mst[visited\_count - 1] = m;

cost += m.wt;

for (int i = 0; i < city\_count; i++) {

if (adj\_mat[m.end][i] != INT32\_MAX) {

edge e;

e.start = m.end;

e.end = i;

e.wt = adj\_mat[e.start][i];

add\_to\_list(adj, e);

}

}

visited[m.end] = 1;

visited\_count++;

}

}

}

void graph::display\_mst() {

cout << "Most efficient network is:\t" << endl;

for (int i = 0; i < city\_count - 1; i++) {

cout << city\_names[mst[i].start] << " to " << city\_names[mst[i].end]

<< " of weight " << mst[i].wt << endl;

}

cout << endl << "The cost of network is:\t" << cost << endl;

}

int main() {

// prims algo

graph g;

int start;

cout << "Enter beginning city:\t";

cin >> start;

start = (start > MAX\_NUM\_CITIES - 1) ? 0 : start;

g.prims\_algo(start);

g.display\_mst();

return 0;

}