**Practical GroupC\_15**

**Aim:**

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.

**Code:**

#include<iostream>

using namespace std;

#define ROW 10

#define COL 10

#define infi 9999

class prims {

int graph[ROW][COL], nodes;

public:

void createGraph();

void primsAlgo();

};

void prims::createGraph() {

int i, j;

cout<< "Enter Total Offices: ";

cin >> nodes;

cout << "\nEnter Adjacency Matrix: \n";

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

for (j = i; j < nodes; j++) {

cout << "Enter distance between " << i << " and " << j << endl;

cin >> graph[i][j];

graph[j][i] = graph[i][j];

}

}

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

for (j = 0; j < nodes; j++) {

if (graph[i][j] == 0)

graph[i][j] = infi; //fill infinity where path is not present

}

}

}

void prims::primsAlgo() {

int selected[ROW], i, j, ne=0;

int zero = 0, one = 1, min = 0, x, y;

int cost = 0;

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

selected[i] = zero;

selected[0] = one; //starting vertex is always node-0

while (ne < nodes - 1) {

min = infi;

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

if (selected[i] == one) {

for (j = 0; j < nodes; j++) {

if (selected[j] == zero) {

if (min > graph[i][j]) {

min = graph[i][j];

x = i;

y = j;

}

}

}

}

}

selected[y] = one;

cout << "\n" << x << " --> " << y;

cost += graph[x][y];

ne++;

}

cout << "\nTotal cost is: " << cost << endl;

}

int main() {

prims MST;

cout << "\nPrims Algorithm to connect several offices\n";

MST.createGraph();

MST.primsAlgo();

}