#include <iostream>

#define V 5

using namespace std;

class MyGraph {

public:

int parent[V]; // Array to Store Parent Nodes

void prims(); // Function which Implements Prim's Algorithm

int find(int i); // Function to find parent of a node

void unionSet(int i, int j); // Function to find Union

int kruskal(); // Function which Implements Kruskal's Algorithm

};

int Am[V][V] = {

{0, 2, 0, 6, 0},

{2, 0, 3, 0, 5},

{0, 3, 0, 0, 0},

{6, 0, 0, 0, 0},

{0, 5, 0, 0, 0}

};

//\*\*\*\*\*\*\*\*\*\*\*\*\* PRIM'S ALGORITHM \*\*\*\*\*\*\*\*\*\*\*\*\*//

void MyGraph::prims() {

int key[V];

bool mstSet[V];

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

mstSet[i] = false;

key[i] = 32767;

}

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < V - 1; count++) {

int mini = 32767;

int u;

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

if (mstSet[i] == false && key[i] < mini) {

u = i;

mini = key[i];

}

}

mstSet[u] = true;

for (int v = 0; v < V; v++) {

if (Am[u][v] && mstSet[v] == false && Am[u][v] < key[v]) {

parent[v] = u;

key[v] = Am[u][v];

}

}

}

cout << "Edge \tWeight\n";

for (int i = 1; i < V; i++) {

cout << parent[i] << " - " << i << " \t" << Am[i][parent[i]] << " \n";

}

int sum = 0;

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

sum+=key[i];

}

cout<<"Minimum cost: "<<sum<<endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTION TO FIND PARENT NODE \*\*\*\*\*\*\*\*\*\*\*\*\*//

int MyGraph::find(int i) {

while (parent[i] != i) {

i = parent[i];

}

return i;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\* UNION SET \*\*\*\*\*\*\*\*\*\*\*\*\*//

void MyGraph::unionSet(int i, int j) {

int s1 = find(i);

int s2 = find(j);

if (s1 != s2) {

parent[s1] = s2;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\* KRUSKAL'S ALGORITHM \*\*\*\*\*\*\*\*\*\*\*\*\*//

int MyGraph::kruskal(){

int mincost = 0;

for(int i = 0; i < V; i++)

parent[i] = i;

int edge\_count = 0;

while (edge\_count < V - 1) {

int min = 32767;

int a = -1;

int b = -1;

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

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

if (find(i) != find(j) && Am[i][j] != 0 && Am[i][j] < min) {

min = Am[i][j];

a = i;

b = j;

}

}

}

if (a != -1 && b != -1) {

if (find(a) != find(b)) {

unionSet(a, b);

mincost += min;

cout << a << " - " << b << " " << min<<endl;

edge\_count++;

}

}

}

return mincost;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN FUNCTION \*\*\*\*\*\*\*\*\*\*\*\*\*//

int main() {

MyGraph g;

/\*cout << "Enter the weighted adjacency matrix:\n";

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

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

cin >> g.Am[i][j];

}

}\*/

cout << "\n\n\nApplying Prim's Algorithm:\n";

g.prims();

cout << "\n\n\nApplying Kruskal's Algorithm:\n";

cout << "Edge \tWeight\n";

cout<<"Minimum Cost: "<<g.kruskal();

return 0;

}