**KRUSHKAL’S ALGO**

#include <iostream>

using namespace std;

void krushkals(int n, int g[100][100], int source, int dist[100]) {

int i, v, count = 1, min;

int visited[100];

for (i = 1; i <= n; i++) {

visited[i] = 0;

dist[i] = g[source][i];

}

visited[source] = 1;

dist[source] = 0;

while (count < n) {

min = 9999;

for (i = 1; i <= n; i++) {

if (dist[i] < min && visited[i] == 0) {

min = dist[i];

v = i;

}

}

visited[v] = 1;

count++;

for (i = 1; i <= n; i++) {

if (dist[i] > dist[v] + g[v][i]) {

dist[i] = dist[v] + g[v][i];

}

}

}

}

int main() {

int g[100][100], source, n, dist[100], i, j;

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

cin >> n;

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

cout << i << "-" << j << ": ";

cin >> g[i][j];

if (g[i][j] == 0 && i != j) {

g[i][j] = 9999; // Representing infinity

}

}

}

cout << "Enter the source vertex: ";

cin >> source;

krushkals(n, g, source, dist);

cout << "The shortest path from source " << source << " is:" << endl;

for (i = 1; i <= n; i++) {

if (source != i) {

cout << source << " -> " << i << " = " << dist[i] << endl;

}

}

return 0;

}

**PRIMS ALGO**

#include <iostream>

#include <climits>

#include <vector>

using namespace std;

int minKey(const vector<int>& key, const vector<bool>& mstSet) {

int min = INT\_MAX, min\_index;

for (int v = 0; v < key.size(); v++) {

if (!mstSet[v] && key[v] < min) {

min = key[v];

min\_index = v;

}

}

return min\_index;

}

void printMST(const vector<int>& parent, const vector< vector<int> >& graph) {

cout << "Edge \tWeight\n";

for (int i = 1; i < parent.size(); i++) {

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

}

}

void primMST(const vector< vector<int> >& graph) {

int V = graph.size();

vector<int> parent(V, -1);

vector<int> key(V, INT\_MAX);

vector<bool> mstSet(V, false);

key[0] = 0; // Initialize key value as 0 for the first vertex

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

int u = minKey(key, mstSet);

mstSet[u] = true;

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

if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {

parent[v] = u;

key[v] = graph[u][v];

}

}

}

printMST(parent, graph);

}

int main() {

int V;

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

cin >> V;

vector< vector<int> > graph(V, vector<int>(V));

cout << "Enter the adjacency matrix for the graph:\n";

}