Lab Experiment No.8

-----------------------------------------------------------------------------------------------------------------------------

Name: Prathamesh Sadashiv Gadekar

Roll No.:14

Class : SE-IT

-----------------------------------------------------------------------------------------------------------------------------

Represent a graph of city using adjacency matrix /adjacency list. Nodes should represent the various

landmarks and links should represent the distance between them. Find the shortest path using Dijkstra's algorithm from single source to all destination.

-----------------------------------------------------------------------------------------------------------------------------

#include <limits.h>

#include <iostream>

using namespace std;

// Number of vertices in the graph means various landmarks in the city of Pune

#define V 9

string landmark[10]={"Aundh","Baner","Wakad","Pimpri","Deccan","Bavdhan","Waraje","Vadgaon","Kothrud",};

// A utility function to find the vertex with minimum

// distance value, from the set of vertices not yet included

// in shortest path tree

int minDistance(int dist[], bool sptSet[])

{ // Initialize min value

int min = INT\_MAX, min\_index;

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

if (sptSet[v] == false && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

// A utility function to print the constructed distance

// array

void printSolution(int dist[], int n)

{

cout<<"\nVertex Distance from Source\n";

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

cout<<landmark[i]<<"\t\t"<<dist[i]<<"\n";

}

void dijkstra(int graph[V][V], int src)

{

int dist[V];

bool sptSet[V]; // sptSet[i] will be true if vertex i is

// included in shortest path tree or shortest distance from src to i is finalized

// Initialize all distances as INFINITE and stpSet[] as false

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

dist[i] = INT\_MAX, sptSet[i] = false;

// Distance of source vertex from itself is always 0

dist[src] = 0;

// Find shortest path for all vertices

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

int u = minDistance(dist, sptSet);

// Mark the picked vertex as processed

sptSet[u] = true;

// Update dist value of the adjacent vertices of the picked vertex.

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

{

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u] + graph[u][v] < dist[v])

dist[v] = dist[u] + graph[u][v];

}

}

// print the constructed distance array

printSolution(dist, V);

}

int main()

{

int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

dijkstra(graph, 2);

return 0;

}

**OUTPUT :**

Vertex Distance from Source

Aundh 12

Baner 8

Wakad 0

Pimpri 7

Deccan 14

Bavdhan 4

Waraje 6

Vadgaon 7

Kothrud 2