Northern Virginia Community College

Ahmad Malik

CSC 202

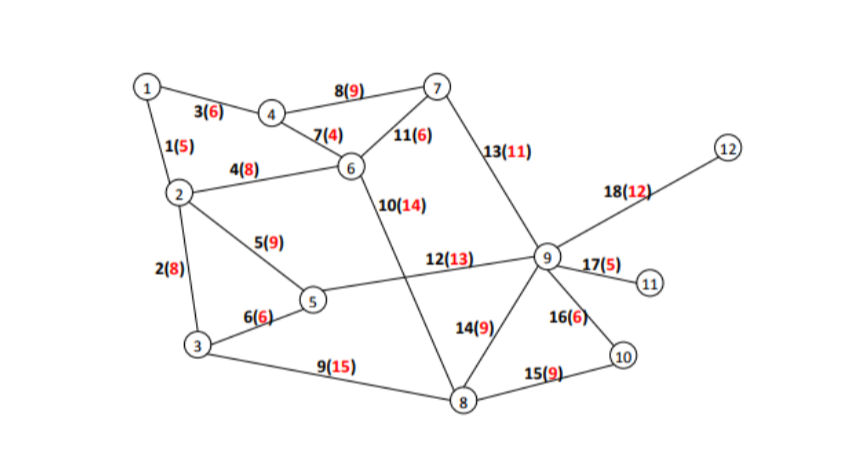
Programming

Project # 2

Project:

Dijkstra’s Algorithm Implementation: Implement an algorithm developed by Edsger Dijkstra (1930-2002) to solve the network shortest path problem in this lab.

Graph:



Translation of graph to 12x12 matrix:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 0 | 5 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 5 | 0 | 8 | 0 | 9 | 8 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 8 | 0 | 0 | 6 | 0 | 0 | 15 | 0 | 0 | 0 | 0 |
| 4 | 6 | 0 | 0 | 0 | 0 | 4 | 9 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 9 | 6 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 0 | 0 |
| 6 | 0 | 8 | 0 | 4 | 0 | 0 | 6 | 14 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 9 | 0 | 6 | 0 | 0 | 11 | 0 | 0 | 0 |
| 8 | 0 | 0 | 15 | 0 | 0 | 14 | 0 | 0 | 9 | 9 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 13 | 0 | 11 | 9 | 0 | 6 | 5 | 12 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 6 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 0 | 0 |

JAVA Source Code:

Class Main:

public class Main

{

public static void main (String[] args)

{

int graph[][] = new int[][]{

{0, 5, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0},

{5, 0, 8, 0, 9, 8, 0, 0, 0, 0, 0, 0},

{0, 8, 0, 0, 6, 0, 0, 15,0, 0, 0, 0},

{6, 0, 0, 0, 0, 4, 9, 0, 0, 0, 0, 0},

{0, 9, 6, 0, 0, 0, 0, 0, 13,0, 0, 0},

{0, 8, 0, 4, 0, 0, 6, 14,0, 0, 0, 0},

{0, 0, 0, 9, 0, 6, 0, 0, 11,0, 0, 0},

{0, 0, 15, 0, 0,14, 0,0, 9, 9, 0, 0},

{0, 0, 0, 0, 13,0, 11,9, 0, 6, 5, 12},

{0, 0, 0, 0, 0, 0, 0, 9, 6, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 12,0, 0, 0},

};

ShortestDistance Y = new ShortestDistance();

Y.algorithm(graph, 0);

ShortestPath Z = new ShortestPath();

Z.algorithm(graph);

}

}

Class ShortestDistance:

class ShortestDistance

{

static final int X=12;

void algorithm (int graph[][], int value)

{

int Dist[] = new int[X];

Boolean Var[] = new Boolean[X];

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

{

Dist[i] = Integer.MAX\_VALUE;

Var[i] = false;

}

Dist[value] = 0;

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

{

int u = minDist(Dist, Var);

Var[u] = true;

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

if (!Var[v] && graph[u][v]!=0 && Dist[u] != Integer.MAX\_VALUE && Dist[u]+graph[u][v] < Dist[v])

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

}

printSol(Dist, X);

}

int minDist(int Dist[], Boolean Var[])

{

int min = Integer.MAX\_VALUE, min\_index=-1;

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

if (Var[v] == false && Dist[v] <= min)

{

min = Dist[v];

min\_index = v;

}

return min\_index;

}

void printSol(int Dist[], int n)

{

System.out.println("Vertex distance from Source");

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

System.out.println("Shortest distance from Node 1 to Node "+(i+1)+" is");

System.out.println(Dist[i]);

}

}

}

Class ShortestPath:

public class ShortestPath{

public void algorithm(int graph[][])

{

int[] Var = new int[12];

int min = 999, nxtNode = 0;

int[] dist = new int[12];

int[] visited = new int[12];

for (int i = 0; i < 12; i++) {

visited[i] = 0;

Var[i] = 0;

for (int j = 0; j < dist.length; j++) {

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

graph[i][j] = 999;

}}}

dist = graph[0];

visited[0] = 1;

dist[0] = 0;

for (int counter = 0; counter < 12; counter++) {

min = 999;

for (int i = 0; i < 12; i++) {

if (min > dist[i] && visited[i] != 1) {

min = dist[i];

nxtNode = i;

}}

visited[nxtNode] = 1;

for (int i = 0; i < 12; i++) {

if (visited[i] != 1) {

if (min + graph[nxtNode][i] < dist[i]) {

dist[i] = min + graph[nxtNode][i];

Var[i] = nxtNode;

}

}

}

}

int j;

for (int i = 0; i < 12; i++) {

if (i != 0) {

System.out.println("Path to "+(i+1)+":");

System.out.print("Path = " + (i + 1));

j = i;

do {

j = Var[j];

System.out.print(" <- " + (j + 1));

} while (j != 0);

System.out.println("");

}

System.out.println();

}

}

}

Output Of Program:

