21BDS0340

Abhinav Dinesh Srivatsa

Computer Networks Lab

Assignment – VI

**Question 1**

Aim:

To implement the distance vector routing algorithm.

Code:

import java.util.Arrays;

class DistanceVectorRouting

{

private

static final int INF = Integer.MAX\_VALUE;

private

int V;

private

int[][] graph;

public

DistanceVectorRouting(int V)

{

this.V = V;

this.graph = new int[V][V];

}

public

void addEdge(int u, int v, int weight)

{

this.graph[u][v] = weight;

this.graph[v][u] = weight;

}

public

void printSolution(int[] dist)

{

System.out.println("Vertex \t Distance from Source");

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

{

System.out.println(node + "\t\t" + dist[node]);

}

}

public

int minDistance(int[] dist, boolean[] visited)

{

int minDist = INF;

int minIndex = 0;

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

{

if (!visited[v] && dist[v] < minDist)

{

minDist = dist[v];

minIndex = v;

}

}

return minIndex;

}

public

void dijkstra(int src)

{

int[] dist = new int[V];

boolean[] visited = new boolean[V];

Arrays.fill(dist, INF);

dist[src] = 0;

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

{

int u = minDistance(dist, visited);

visited[u] = true;

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

{

if (!visited[v] && graph[u][v] != 0 && dist[u] != INF &&

dist[v] > dist[u] + graph[u][v])

{

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

}

}

}

printSolution(dist);

}

public

static void main(String[] args)

{

DistanceVectorRouting g = new DistanceVectorRouting(9);

g.addEdge(0, 1, 4);

g.addEdge(0, 7, 8);

g.addEdge(1, 2, 8);

g.addEdge(1, 7, 11);

g.addEdge(2, 3, 7);

g.addEdge(2, 8, 2);

g.addEdge(2, 5, 4);

g.addEdge(3, 4, 9);

g.addEdge(3, 5, 14);

g.addEdge(4, 5, 10);

g.addEdge(5, 6, 2);

g.addEdge(6, 7, 1);

g.addEdge(6, 8, 6);

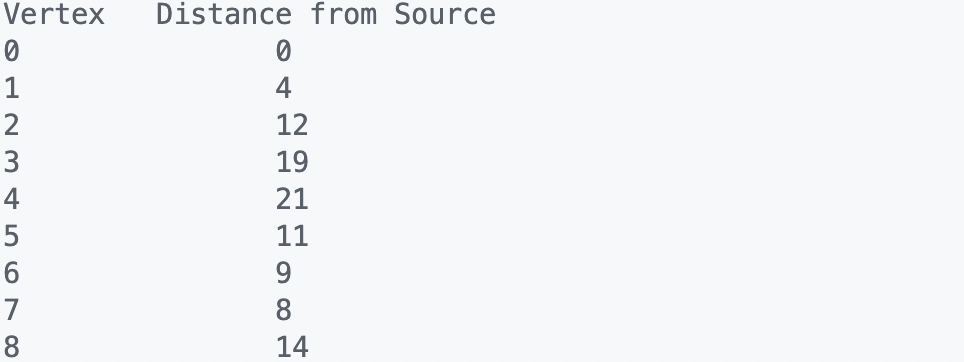
g.addEdge(7, 8, 7);

g.dijkstra(0);

}

}

Output:



**Question 2**

Aim:

To implement the link state routing algorithm.

Code:

import java.util.Arrays;

class LinkStateRouting {

private static final int INF = Integer.MAX\_VALUE;

private int V;

private int[][] graph;

public LinkStateRouting(int V) {

this.V = V;

this.graph = new int[V][V];

}

public void addEdge(int u, int v, int weight) {

this.graph[u][v] = weight;

this.graph[v][u] = weight;

}

public void printSolution(int[] dist) {

System.out.println("Vertex \t Distance from Source");

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

System.out.println(node + "\t\t" + dist[node]);

}

}

public int minDistance(int[] dist, boolean[] visited) {

int minDist = INF;

int minIndex = 0;

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

if (!visited[v] && dist[v] < minDist) {

minDist = dist[v];

minIndex = v;

}

}

return minIndex;

}

public void dijkstra(int src) {

int[] dist = new int[V];

boolean[] visited = new boolean[V];

Arrays.fill(dist, INF);

dist[src] = 0;

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

int u = minDistance(dist, visited);

visited[u] = true;

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

if (!visited[v] && graph[u][v] != 0 && dist[u] != INF &&

dist[v] > dist[u] + graph[u][v]) {

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

}

}

}

printSolution(dist);

}

public static void main(String[] args) {

LinkStateRouting g = new LinkStateRouting(9);

g.addEdge(0, 1, 3);

g.addEdge(0, 7, 8);

g.addEdge(1, 2, 24);

g.addEdge(1, 7, 11);

g.addEdge(2, 3, 6);

g.addEdge(2, 8, 2);

g.addEdge(2, 5, 4);

g.addEdge(3, 4, 1);

g.addEdge(3, 5, 14);

g.addEdge(4, 5, 0);

g.addEdge(5, 6, 2);

g.addEdge(6, 7, 1);

g.addEdge(6, 8, 6);

g.addEdge(7, 8, 9);

g.dijkstra(0);

}

}

Output:

A screenshot of a computer

Description automatically generated