

# Bansilal Ramnath Agarwal Charitable Trust’s

# Vishwakarma Institute of Technology

#### (An Autonomous Institute affiliated to Savitribai Phule Pune University)

**COMPUTER NETWORKS LAB ASSIGNMENTS**

|  |  |
| --- | --- |
| NAME | ABBAS MADHVASWALA |
| PRN | 12110285 |
| DIVISION | CS-C |
| ROLL.NO | 13 |
| BATCH | B3 |

**LAB ASSIGNMENT NO : 04**

**PROBLEM STATEMENT**

**Problem Statement:** Write a program to find the shortest path using Dijkstra Equation for Link State Routing Protocol which is used by Open Shortest Path First Protocol (OSPF) in the Internet for the network flow provided by instructor.

# Program Code –

*import* java.util.\*;

*public class* OSPF\_Dijkstra {

*private int* dist[];

*private Set*<Integer> settled; *private* PriorityQueue<Node> pq; *private int* V;

*List*<*List*<Node> > adj;

*public* OSPF\_Dijkstra(*int* V){

*this*.V = V;

dist = *new int*[V];

settled = *new* HashSet<Integer>();

pq = *new* PriorityQueue<Node>(V, *new* Node());

}

*public void* dijkstra(*List*<*List*<Node> > adj, *int* src){

*this*.adj = adj;

*for* (*int* i = 0; i < V; i++){ dist[i] = Integer.***MAX\_VALUE***;

}

pq.add(*new* Node(src, 0));

dist[src] = 0;

*while* (settled.size() != V) {

*if* (pq.isEmpty()){

*return*;

}

*int* u = pq.remove().node;

*if* (settled.contains(u)){

*continue*;

}

settled.add(u); e\_Neighbours(u);

}

}

*private void* e\_Neighbours(*int* u){

*int* edgeDistance = -1;

*int* newDistance = -1;

*for* (*int* i = 0; i < adj.get(u).size(); i++) { Node v = adj.get(u).get(i);

*if* (!settled.contains(v.node)) { edgeDistance = v.cost;

newDistance = dist[u] + edgeDistance;

*if* (newDistance < dist[v.node]){ dist[v.node] = newDistance;

}

pq.add(*new* Node(v.node, dist[v.node]));

}

}

}

*public static void* main(String arg[]){

*int* V = 5;

*int* source = 0;

*List*<*List*<Node> > adj = *new* ArrayList<*List*<Node> >();

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

*List*<Node> item = *new* ArrayList<Node>(); adj.add(item);

}

adj.get(0).add(*new* Node(1, 9));

adj.get(0).add(*new* Node(2, 6));

adj.get(0).add(*new* Node(3, 5));

adj.get(0).add(*new* Node(4, 3));

adj.get(2).add(*new* Node(1, 2));

adj.get(2).add(*new* Node(3, 4)); OSPF\_Dijkstra dpq = *new* OSPF\_Dijkstra(V); dpq.dijkstra(adj, source);

System.***out***.println("The shortest path from source Router to destination Routers:");

*for* (*int* i = 0; i < dpq.dist.length; i++){ System.***out***.println("Router "+ source + " to Router " + i + " is

" +

dpq.dist[i]);

}

}

}

*class* Node *implements Comparator*<Node> {

*public int* node; *public int* cost; *public* Node() {}

*public* Node(*int* node, *int* cost){

*this*.node = node;

*this*.cost = cost;

}

@Override *public int* compare(Node node1, Node node2){

*if* (node1.cost < node2.cost)

*return* -1;

*if* (node1.cost > node2.cost)

*return* 1;

*return* 0;

}

}

# Output –

