Name:Ankita Sanjay Kakade

Class: AI-B

Roll No.:19

Subject: Computer Network

**Assignment No.19**

**Problem Statement:** Write a program to implement:

1. Network Routing: Shortest path routing, AODV

**Code:** Shortest Path Routing

import java.util.\*;

public class ShortestPathRouting {

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

    public static void dijkstra(int[][] graph, int src) {

        int V = graph.length;

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

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

                }

            }

        }

        printSolution(dist);

    }

    private static int minDistance(int[] dist, boolean[] visited) {

        int min = INF;

        int minIndex = -1;

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

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

                min = dist[v];

                minIndex = v;

            }

        }

        return minIndex;

    }

    private static void printSolution(int[] dist) {

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

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

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

        }

    }

    public static void main(String[] args) {

        int[][] graph = {

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

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

            // Add the rest of the adjacency matrix here

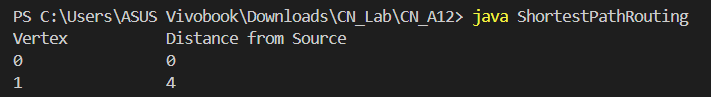
        };

        dijkstra(graph, 0);

    }

}

**Output:**



**Code 2:** AODV

import java.util.\*;

class Node {

    int id;

    HashMap<Integer, Integer> routingTable; // Destination -> Next hop

    Node(int id) {

        this.id = id;

        routingTable = new HashMap<>();

    }

}

public class AODV {

    HashMap<Integer, Node> network;

    public AODV() {

        network = new HashMap<>();

    }

    // Function to add a node to the network

    public void addNode(int id) {

        if (!network.containsKey(id)) {

            network.put(id, new Node(id));

        }

    }

    // Function to initialize route discovery

    public void routeDiscovery(int source, int destination) {

        if (!network.containsKey(source) || !network.containsKey(destination)) {

            System.out.println("Source or destination node does not exist.");

            return;

        }

        // Simulate route discovery process

        Node sourceNode = network.get(source);

        Node destinationNode = network.get(destination);

        if (sourceNode.routingTable.containsKey(destination)) {

            System.out.println("Route already exists.");

            return;

        }

        // Simulate route discovery by flooding RREQ (Route Request) packets

        System.out.println("Route discovery initiated from " + source + " to " + destination);

        System.out.println("Sending Route Request (RREQ) from " + source + " to its neighbors...");

        // Forward RREQ packets to neighbors (simulation)

        for (int neighborId : sourceNode.routingTable.keySet()) {

            // Simulate packet transmission to neighbors

            System.out.println("Forwarding RREQ from " + source + " to neighbor " + neighborId);

            // Here you would actually send RREQ packets to neighbors over the network

        }

        // Upon receiving RREQ, neighbors process it and may forward to their neighbors

        // Eventually, RREQ reaches destination or an intermediate node with a route to the destination

        // Simulate RREP (Route Reply) from destination to source

        System.out.println("Destination " + destination + " reached.");

        System.out.println("Sending Route Reply (RREP) from " + destination + " to " + source);

        // Upon receiving RREP, intermediate nodes update their routing tables

        // Update routing table at source

        sourceNode.routingTable.put(destination, destination);

        System.out.println("Route established from " + source + " to " + destination);

    }

    public static void main(String[] args) {

        AODV aodv = new AODV();

        // Create nodes and add to the network

        aodv.addNode(1);

        aodv.addNode(2);

        aodv.addNode(3);

        // Add more nodes as needed

        // Simulate route discovery from node 1 to node 3

        aodv.routeDiscovery(1, 3);

    }

}

**Output:**

