# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(Autonomous Institute of Savitribai Phule Pune University)*



**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | **CS SY B** |
| **Batch** | **3** |
| **GR no.** | **12320165** |
| **Roll no.** | **83** |
| **Name** | **Komal Mahadev Potdar** |

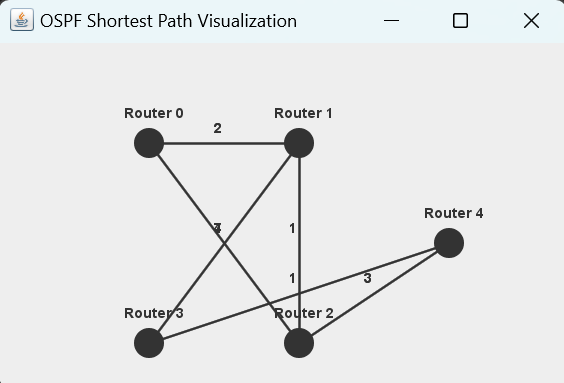
**Assignment No. 5**

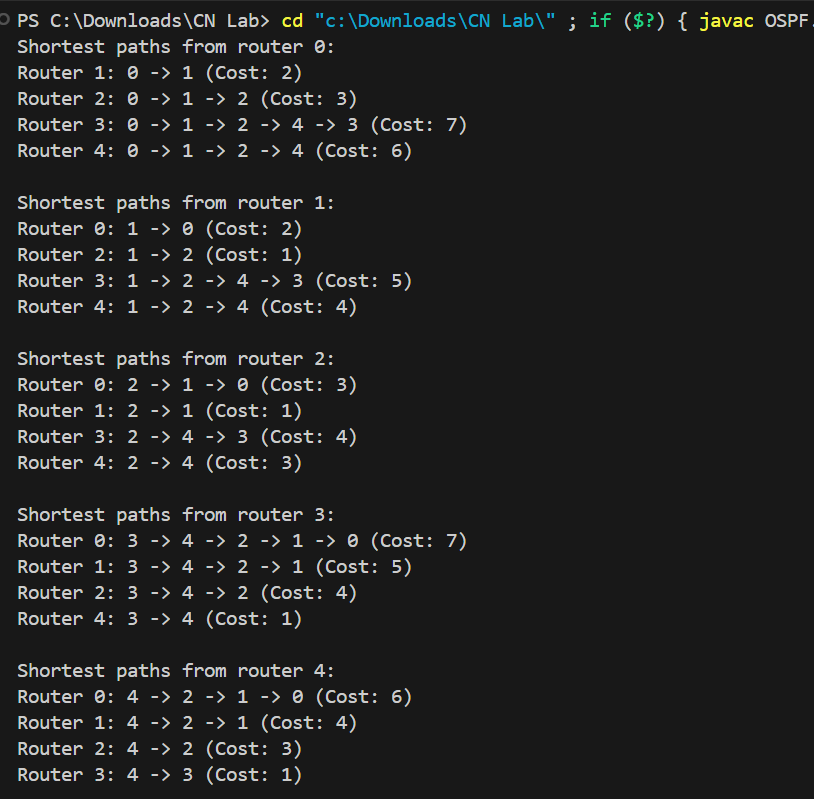
**Title:** Write a program to find the shortest path using Dijkstra Equation for Link State Routing Protocol which is used by Open Shortest Path First Protocol (OSPF) in the Internet for the network flow provided by instructor.

**Code:**

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.util.ArrayList;  import java.util.Arrays;  import java.util.PriorityQueue;  import java.util.List;  class Edge {  int targetNode;  int weight;  Edge(int targetNode, int weight) {  this.targetNode = targetNode;  this.weight = weight;  }  }  class Node implements Comparable<Node> {  int id;  int distance;  Node(int id, int distance) {  this.id = id;  this.distance = distance;  }  @Override  public int compareTo(Node other) {  return Integer.compare(this.distance, other.distance);  }  }  public class OSPF extends JPanel {  private final int numRouters;  private final List<List<Edge>> adjList;  private final int[][] routerPositions;  public OSPF(int numRouters) {  this.numRouters = numRouters;  adjList = new ArrayList<>(numRouters);  for (int i = 0; i < numRouters; i++) {  adjList.add(new ArrayList<>());  }  // Define positions for routers in the GUI  routerPositions = new int[][]{  {150, 100}, // Router 0  {300, 100}, // Router 1  {300, 300}, // Router 2  {150, 300}, // Router 3  {450, 200} // Router 4  };  }  public void addLink(int srcRouter, int destRouter, int cost) {  adjList.get(srcRouter).add(new Edge(destRouter, cost));  adjList.get(destRouter).add(new Edge(srcRouter, cost));  }  public void computeShortestPaths(int sourceRouter) {  int[] distances = new int[numRouters];  int[] previous = new int[numRouters];  Arrays.fill(distances, Integer.MAX\_VALUE);  Arrays.fill(previous, -1);  distances[sourceRouter] = 0;  PriorityQueue<Node> pq = new PriorityQueue<>();  pq.add(new Node(sourceRouter, 0));  while (!pq.isEmpty()) {  Node currentNode = pq.poll();  int currentRouter = currentNode.id;  for (Edge edge : adjList.get(currentRouter)) {  int targetRouter = edge.targetNode;  int newDist = distances[currentRouter] + edge.weight;  if (newDist < distances[targetRouter]) {  distances[targetRouter] = newDist;  previous[targetRouter] = currentRouter;  pq.add(new Node(targetRouter, newDist));  }  }  }  System.out.println("Shortest paths from router " + sourceRouter + ":");  for (int i = 0; i < numRouters; i++) {  if (i != sourceRouter) {  System.out.print("Router " + i + ": ");  printPath(i, previous);  System.out.println(" (Cost: " + distances[i] + ")");  }  }  System.out.println();  // Trigger repaint to update the GUI with the computed shortest paths  repaint();  }  private void printPath(int router, int[] previous) {  if (previous[router] == -1) {  System.out.print(router);  return;  }  printPath(previous[router], previous);  System.out.print(" -> " + router);  }  @Override  protected void paintComponent(Graphics g) {  super.paintComponent(g);  Graphics2D g2d = (Graphics2D) g;  g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);  // Set a larger font for better visibility of the costs  g2d.setFont(new Font("Arial", Font.BOLD, 14));  // Draw routers  for (int i = 0; i < numRouters; i++) {  int[] pos = routerPositions[i];  g2d.fillOval(pos[0] - 15, pos[1] - 15, 30, 30);  g2d.drawString("Router " + i, pos[0] - 25, pos[1] - 25);  }  // Draw connections  g2d.setStroke(new BasicStroke(2));  for (int i = 0; i < numRouters; i++) {  for (Edge edge : adjList.get(i)) {  int[] pos1 = routerPositions[i];  int[] pos2 = routerPositions[edge.targetNode];  g2d.drawLine(pos1[0], pos1[1], pos2[0], pos2[1]);  // Adjust the position of the cost label to be clearly visible  int midX = (pos1[0] + pos2[0]) / 2;  int midY = (pos1[1] + pos2[1]) / 2;  g2d.drawString(String.valueOf(edge.weight), midX - 10, midY - 10);  }  }  }  public static void main(String[] args) {  OSPF ospf = new OSPF(5);  ospf.addLink(0, 1, 2);  ospf.addLink(0, 2, 4);  ospf.addLink(1, 2, 1);  ospf.addLink(1, 3, 7);  ospf.addLink(2, 4, 3);  ospf.addLink(3, 4, 1);  // Display the GUI  JFrame frame = new JFrame("OSPF Shortest Path Visualization");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(600, 400);  frame.add(ospf);  frame.setVisible(true);  // Compute and display shortest paths for each router  for (int i = 0; i < 5; i++) {  ospf.computeShortestPaths(i);  }  }  } |

**Output:**

****

****