**Ramdeobaba University, Nagpur**

**Department of Computer Science and Engineering**

**Session: 2025-26**

**Design and Analysis of Algorithms Lab III Semester**

**PRACTICAL NO. 6**

**Yash Wanjari**

**A2\_B3\_40**

**Aim:** Construction of Single Source Shortest Path

**Problem Statement:** Develop a system to optimize the delivery routes for a fleet of vehicles in a metropolitan area. The system should efficiently calculate the shortest paths between multiple pickup and delivery points, taking into account traffic congestion and road conditions.

Implement the Bellman-Ford algorithm to find the shortest path from a central depot to each delivery location while considering varying transportation costs and time constraints.

Consider the following criteria for determining connections within the same state in India:

i. Determine the latitude and longitude of addresses within the same city. Select 6 to 8 addresses, with one designated as zero mile, and construct a fully connected graph.

ii. Designate the zero-mile location as the pickup point.

iii. Calculate the shortest paths between the pickup point and delivery points.

Code:

import java.util.\*;

class Edge

{

int src,des;

double wt;

Edge(int s,int d,double w)

{

src = s;

des = d;

wt = w;

}

}

public class deliveryRoutes

{

public static double distFormula(double lat1, double lon1, double lat2, double lon2)

{

double dLat = lat2 - lat1;

double dLon = lon2 - lon1;

double dist = Math.sqrt(Math.pow(dLat, 2) + Math.pow(dLon, 2)) \* 111;

return dist;

}

static void bellmanFord(List<Edge> edges, int V, int src)

{

double[] dist = new double[V];

Arrays.fill(dist, Double.MAX\_VALUE);

dist[src] = 0;

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

{

for (Edge e : edges)

{

if (dist[e.src] != Double.MAX\_VALUE && dist[e.src] + e.wt < dist[e.des])

{

dist[e.des] = dist[e.src] + e.wt;

}

}

}

for (Edge e : edges)

{

if (dist[e.src] != Double.MAX\_VALUE && dist[e.src] + e.wt < dist[e.des])

{

System.out.println("Negative weight cycle detected");

return;

}

}

System.out.println("Shortest distances from pickup point " + src + ":");

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

{

if (dist[i] == Double.MAX\_VALUE)

{

System.out.println("to loc " + i + " : Unreachable");

}else{

System.out.printf("to loc %d : %.2f km\n",i,dist[i]);

}

}

}

public static void main(String[] args)

{

int V = 6;

int src = 0;

double[][] locations = {

{21.1458, 79.0882}, // Nagpur

{19.0760, 72.8777}, // Mumbai

{18.5204, 73.8567}, // Pune

{19.9975, 73.7898}, // Nashik

{20.7090, 77.0080}, // Akola

{20.9333, 77.7500} // Amravati

};

List<Edge> edges = new ArrayList<>();

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

{

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

{

if(i != j)

{

double dist = distFormula(locations[i][0], locations[i][1], locations[j][0], locations[j][1]);

edges.add(new Edge(i, j, dist));

}

}

}

bellmanFord(edges,V,src);

}

}

*O/p:*

