Actividad 2 Una empresa de logística tiene varios centros de distribución en diferentes ciudades de una región y necesita optimizar las rutas de entrega de sus camiones. Cada centro de distribución está conectado a otros centros mediante carreteras, y cada carretera tiene un tiempo de viaje asociado en minutos. Además, algunos centros pueden tener costos adicionales asociados que pueden resultar en tiempos negativos en algunas rutas debido a descuentos especiales o condiciones excepcionales. La empresa desea no solo minimizar el tiempo total de entrega desde su centro de distribución principal hasta todas las otras ciudades, sino también identificar si existen ciclos negativos que podrían llevar a oportunidades de ahorro infinito en el sistema de rutas. Objetivo: Aplicar el algoritmo de Floyd-Warshall para: Encontrar el tiempo mínimo de entrega desde el centro de distribución principal hasta todos los demás centros de distribución, considerando las diferentes rutas disponibles.

import java.util.Arrays;

class OptimizacionRutas {

static final int INF = 99999; // Representación de infinito (para rutas no existentes)

public static void floydWarshall(int[][] tiempos, int numCentros) {

int[][] distancias = new int[numCentros][numCentros];

// Copiar la matriz de tiempos en la matriz de distancias

for (int i = 0; i < numCentros; i++) {

System.arraycopy(tiempos[i], 0, distancias[i], 0, numCentros);

}

// Aplicar el algoritmo de Floyd-Warshall

for (int k = 0; k < numCentros; k++) {

for (int i = 0; i < numCentros; i++) {

for (int j = 0; j < numCentros; j++) {

if (distancias[i][k] != INF && distancias[k][j] != INF) {

distancias[i][j] = Math.min(distancias[i][j], distancias[i][k] + distancias[k][j]);

}

}

}

}

// Detectar ciclos negativos

for (int i = 0; i < numCentros; i++) {

if (distancias[i][i] < 0) {

System.out.println("⚠️ Se ha detectado un ciclo negativo en el centro " + (i + 1));

return;

}

}

// Mostrar la matriz de distancias mínimas

System.out.println("🚚 Tiempos mínimos de entrega entre centros de distribución:");

imprimirMatriz(distancias, numCentros);

}

public static void imprimirMatriz(int[][] matriz, int numCentros) {

for (int i = 0; i < numCentros; i++) {

for (int j = 0; j < numCentros; j++) {

if (matriz[i][j] == INF) {

System.out.print("INF\t");

} else {

System.out.print(matriz[i][j] + "\t");

}

}

System.out.println();

}

}

public static void main(String[] args) {

int numCentros = 4; // Número de centros de distribución

// Matriz de tiempos (INF si no hay conexión)

int[][] tiempos = {

{0, 5, INF, 10},

{INF, 0, 3, INF},

{INF, INF, 0, 1},

{INF, INF, INF, 0}

};

floydWarshall(tiempos, numCentros);

}

}