Laboratory work #7

Student: CAO Xinyang  
Student ID: 20321308  
Timus Name: hduads2022\_20321308

Mail: c.x\_yang@foxmail.com

Problem #1450

Screenshot from Timus:



Explanation of algorithm:

Use the Bellman-Ford algorithm. First, initialize all points, setting the distance from the origin to 0 and all other points to -1. Then, we loop over all the points except the origin, and we loop over all the edges inside the loop, and we do the relaxation. If the final result is -1, the result does not exist.

Computational complexity of algorithm:

T(N) = O(M \* N)

Source code:

import java.io.\*;

import java.util.Arrays;

public class russianPipeline {

static int max = 1000;

static long[][] value = new long[max][max];

static long[] result = new long[max];

static boolean[] done = new boolean[max];

static int n, m, s, f;

public static void main(String[] args) throws IOException {

StreamTokenizer in = new StreamTokenizer(new BufferedReader(new InputStreamReader(System.in)));

int a, b, c;

in.nextToken();

n = (int)in.nval;

in.nextToken();

m = (int)in.nval;

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

in.nextToken();

a = (int)in.nval;

in.nextToken();

b = (int)in.nval;

in.nextToken();

c = (int)in.nval;

value[a][b] = c;

}

in.nextToken();

s = (int)in.nval;

in.nextToken();

f = (int)in.nval;

Arrays.fill(done, false);

Arrays.fill(result, -1);

done[s] = true;

result[s] = 0;

getLongestPaths(f);

if (result[f] < 0) {

System.out.println("No solution");

} else {

System.out.println(result[f]);

}

}

public static long getLongestPaths(int f) {

if (done[f]) {

return result[f];

}

for (int i = 1; i <= n; i++) {

if (value[i][f] != 0) {

result[f] = Math.max(result[f], getLongestPaths(i) + value[i][f]);

}

}

done[f] = true;

if (result[f] < 0) {

result[f] = Integer.MIN\_VALUE;

}

return result[f];

}

}