Laboratory work # 7

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114
Mail: jhlxhrq@163.com

Problem # 1450. Russian Pipelines

Screenshot from Timus:

9874360	13:44:31 15 May 2022	hduads2022_20321114	1450. Russian Pipelines	Java 1.8	Accepted		0.171	8 476 KB

Explanation of algorithm:

1. Sort each edge topologically and sum up each event, data and resolution.

Computational complexity of algorithm:

$$O(N+M)$$

Source code:

```java
import java.io.*;
import java.util.Arrays;

public class RussianPipes {
    public static void main(String[] args) throws IOException {
        new RussianPipes().run();
    }

    StreamTokenizer in;
    PrintWriter out;

    static final int MAXSIZE = 1000;
    long[][] event = new long[MAXSIZE][MAXSIZE];
    long[] data = new long[MAXSIZE];
    boolean[] isAccessible = new boolean[MAXSIZE];
    int M, N;
    int startPoint, destPoint;

    int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;
    }
```

```
    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
        out = new PrintWriter(System.out);
        solve();
        out.flush();
    }

    void solve() throws IOException {
        prepareDate();
        startPoint = nextInt();
        destPoint = nextInt();
        Arrays.fill(isAccessible, false);
        Arrays.fill(data, -1);
        isAccessible[startPoint] = true;
        data[startPoint] = 0;
        RESOLVE(destPoint);
        out.println(data[destPoint] < 0 ? "No solution" : data[destPoint]);
    }

    void prepareDate() throws IOException {
        int x, y, pipes;
        N = nextInt();
        M = nextInt();
        for (int i = M; i > 0 ; i--) {
            x = nextInt();
            y = nextInt();
            pipes = nextInt();
            event[x][y] = pipes;
        }
    }

    long RESOLVE(int x) {
        if (!isAccessible[x]) {
            for (int i = 1; i <= N; i++) {
                if (event[i][x] != 0) {
                    data[x] = Math.max(data[x], RESOLVE(i) + event[i][x]);
                }
            }
            isAccessible[x] = true;
            if (data[x] < 0) {
                data[x] = Integer.MIN_VALUE;
            }
        }
        return data[x];
    }
}
```