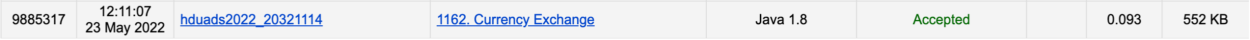
Laboratory work # 8

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1162. Currency Exchange

Screenshot from Timus:



9885317	12:11:07 23 May 2022	hduads2022_20321114	1162. Currency Exchange	Java 1.8	Accepted		0.093	552 KB

Explanation of algorithm:

1. Use the Bellman-Ford algorithms to do the relaxation operation in the graph to find get all possible shortest paths.

Computational complexity of algorithm:

$$O(MN)$$

Source code:

```
import java.io.*;

public class CurrencyExchange {
    StreamTokenizer in;
    PrintWriter out;
    int e, noc, noe, cn;
    double ca;
    Exchange[] ea = new Exchange[1000];
    double[] w = new double[200];

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

    double nextNum() throws IOException {
        in.nextToken();
        return in.nval;
    }

    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
```

```
        out = new PrintWriter(System.out);
        solve();
        out.flush();
    }

    void prepare() throws IOException {
        noc = (int) nextNum();
        noe = (int) nextNum();
        cn = (int) nextNum();
        ca = nextNum();
        int a, b;
        for (int i = 0; i < noe; i++) {
            a = (int) nextNum();
            b = (int) nextNum();
            add(a, b, nextNum(), nextNum());
            add(b, a, nextNum(), nextNum());
        }
    }

    void solve() throws IOException {
        prepare();
        out.println(increase() == 1 ? "YES" : "NO");
    }

    void add(int f, int t, double r, double c) {
        ea[++e] = new Exchange(f, t, r, c);
    }

    int increase() {
        w[cn] = ca;
        Exchange ce;
        for (int i = 1; i < noc; i++) {
            boolean isContinue = false;
            double we;
            for (int j = 1; j <= e; j++) {
                ce = ea[j];
                we = (w[ce.getF()] - ce.getC()) * ce.getR();
                if (we > w[ce.getT()]) {
                    isContinue = true;
                    w[ce.getT()] = we;
                }
            }
            if (!isContinue) {
                break;
            }
        }
        for (int i = 1; i <= e; i++) {
            ce = ea[i];
            if ((w[ce.getF()] - ce.getC()) * ce.getR() > w[ce.getT()]) {
                return 1;
            }
        }
        return 0;
    }

}

class Exchange {
    private int f;
    private int t;
    private double r;
    private double c;

    public Exchange(int f, int t, double r, double c) {
        this.f = f;
        this.t = t;
        this.r = r;
        this.c = c;
    }

    public int getF() {
        return f;
    }

    public int getT() {
        return t;
    }

    public double getR() {
        return r;
    }

    public double getC() {
        return c;
    }
}
```