Task: UFL
Uniform flow

Stage CSPC 2006. Day Fourth. Source file `ufl.*` *2006.17.06*
Available memory: 32 MB.

Imagine a system of n junctions, numbered from 1 to n, connected with undirected pipes in which water flows. The junction number one is *source*, the n-th one is *sink*. Every pipe connects exactly two distinct junctions and is assigned one number — its *size*, that is the maximum velocity of water flowing in it. The water flow must adhere to the regular conservation principle: for every junction, except source and sink, the amount of water flowing into the junction is equal to the amount of water flowing out of it.

Now we add one more limitation: for each pair (J_1, J_2) of junctions the sum of water velocities over all the pipes on an arbitrary path from J_1 to J_2 is constant (for this pair of junctions). We calculate the sums in such a way, that if the water flows against direction of the path from J_1 to J_2, its velocity is negated.

You job is to calculate the maximal flow in the network.

Task

Write a program, that:

- reads the description of the network from standard input,
- calculates the maximal flow,
- writes the result on standard output.

Input

The first line of the standard input contains one natural number n — the number of junctions in the network ($2 \leq n \leq 100$). The second line contains one natural number m ($1 \leq m \leq 5\,000$) — the number of pipes in the network. The next m lines contain the descriptions of the pipes. Each pipe is represented by three integers a_i, b_i, c_i. a_i and b_i are the numbers of junctions connected by the pipe and c_i ($0 \leq c_i \leq 10\,000$) is its size. There can be multiple pipes between a pair of junctions.

Output

The first line of standard output should contain exactly one floating point value — the maximal achievable flow. The number must match the exact result with the maximal absolute difference of 10^{-3}.

The SVIO library has no functionality for writing floating point numbers. Do not use SVIO for this task.

Example

For the input data:

```
4
6
1 3 2
1 2 3
1 2 2
2 4 5
2 3 2
3 4 5
```

the correct result is:

```
5.200000
```

Remark

To avoid rounding problems, we suggest you use *long double* (C/C++) or *extended* (Pascal) data types.