Infinity is one of my favorite game series. If you have ever played Never7, you'll know that Haruka, Kurumi and Izumi all like water.



If you have ever played Ever17, you'll know that the entire story of Ever17 happened under the sea.

So "water" is very important in these two games. That's why I made this problem, in which you need to design an amazing water system.

It is a circulation system (i.e. no "source" or "sink" of water) consisting of m pipes connecting n junctions. No water is created or destroyed during circulation, so for each junction, the total amount of water coming into it should be the same of the amount going out of it, for each unit time.

Pipes are deformable, so we can easily control the water speed through them. However, each pipe has a pair of lower-bound and upper-bound, and the actual water speed through it must lie within the bounds.

Pipes are transparent, so you can actually see how fast the water is going through each pipe. To make it look beautiful, the water speed should be as balanced as possible. I.e. the difference between the maximal water speed and the minimal water speed should be minimized.

Could you find the optimal design?

Input

The first line contains the number of test cases T ($T \leq 100$). Each test case begins with two integers n and m ($2 \leq n \leq 50$, $1 \leq m \leq 200$), the number of junctions and the number of pipes. Each of the following m lines contains four integers u, v, b, c ($1 \leq u, v \leq n$, $u \neq v$, $0 \leq b \leq c \leq 100$), that means there is a pipe connecting junction u and v, and the speed of water flowing from u to v, denoted by f, should satisfy $b \leq f \leq c$. Junctions are numbered 1 to n, and pipes are numbered 1 to m (in the same order that they appear in the input). Note that the pipes need not be straight, so two junctions can be connected by several pipes.

Output

For each test case, print the minimal difference between the maximum speed and the minimum speed to five decimal places. If there is no solution, print '`-1`'.

Sample Input

```
3
4 4
1 2 1 4
2 3 2 5
3 4 3 6
4 1 4 7
3 3
1 2 1 2
2 3 2 3
3 1 3 4
2 3
1 2 3 3
2 1 0 10
2 1 0 10
```

Sample Output

```
Case 1: 0.00000
Case 2: -1
Case 3: 1.50000
```