Yesterday, you were working on a strange machine found in an old garage (later, you find out it was *Dr. Emmett Brown*'s garage), when you accidentally pushed a button and were sent back in time to 48 BC with only a laptop in your hand! Fortunately, you knew enough Latin to communicate with one of the locals who turned out to be Caesar's trade advisor. The advisor invited you to his house and gave you food. In return, you asked him how you could help him. He answered that recently, Caesar has decided to establish a new province in a high profit region of his empire. The empire is a network of cities connected by bidirectional roads, and each road has a certain *profit per annum* (PPA) value. The PPA of any given road is calculated by subtracting its annual upkeep cost from its annual revenue.

The new province will consist of a connected, non-empty set of roads (and the corresponding cities). The province must also have the highest possible average PPA. Caesar further asked his trade advisor to determine the maximum possible number of cities in such a province. Now, the advisor needs your help to solve this problem, or he is worried that Caesar will put him to the sword. You decide to write a program to solve this problem, but you should remember that your laptop has a limited remaining battery and you must be fast!

Input

Each test case starts with two integers $1 < n \leq 500$ and $1 \leq m \leq 1,000,000$, the numbers of cities and roads in the empire respectively. The next m lines each contain three integers where the first two are distinct, between 1 and n, and represent the endpoints of a road, while the third integer is the PPA of that road (fits in a signed 32 bit variable). Input is terminated with a line consisting of $n = m = 0$.

Output

There is one line of output for each test case consisting of a single integer that is the maximum number of cities in a highest average PPA province.

Sample Input

```
4 5
1 2 100
1 3 100
1 4 1
2 3 100
3 4 1
9 14
1 2 9
6 9 8
2 4 9
2 3 9
4 5 1
4 3 9
5 9 2
9 8 9
7 8 9
7 9 5
6 7 9
5 6 4
5 8 7
7 5 9
0 0
```

Sample Output

```
3
5
```