Pavers

Program Name: `Paver.java`   **Input File:** `paver.dat`

The municipal chronicals of an unbelievably lordly major town in a land far, far, away tell the following story: once upon a time the newly crowned King Günther decided to visit all towns in his kingdom. The people of the unbelievably lordly major town expected that King Günther would like to see some of the most famous buildings in their town. For the lordly citizens it seemed neccessary that all streets in the town that the king would have to use had to be cobbled with stone. Unfortunately the unbelievably lordly major town had not much money at that time as they used most of their savings to erect the highest cathedral the world had ever seen.

Roumours were afloat that the real reason for their thriftiness was not that the town treasury was empty but that King Günther came to the throne by deceiving his father King Erwin and that in his youth he made a pact with the devil. In any event, the citizens of the unbelievably lordly major town decided to pave only as many streets as were absolutely necessary to reach every major building.

Can you help the citizens of the unbelievably lordly major town to find out which streets should be paved?

It might be useful to know that all major buildings are either at the end of a street or at an intersection. You may also assume that all buildings in the town are connected by the given streets.

Input

t   number of test cases ($1 \le t \le 100$)
p   price to pave one furlong of street (positive integer)
n   number of main buildings in the town ($1 \le n \le 1{,}000$)
m   number of streets in the town ($1 \le m \le 300{,}000$)
a b c   street from building a to building b with length c
        (lengths are given in furlongs and the buildings are numbered from 1 to n)

Output

For each test case output the price of the cheapest possibility to reach all main buildings in the city on paved streets. You can assume that the result will be smaller than 2^{32}.

Sample Input

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

Sample Output

```
12
```