Problems

Problem	TELE	DOG

Source file	`tele.pas`
	`tele.c`
	`tele.cpp`
		`dog.pas`
		`dog.c`
		`dog.cpp`

Input file	`tele.in`	`dog.in`
Output file	`tele.out`	`dog.out`
Time limit per test	10 sekundi	10 sekundi
Number of tests	10	10
Points per test	7	8
Total points	**70**	**80**
	150	

TELE

A TV-network plans to broadcast an important football match. Their network of transmitters and users can be represented as a **tree**. The root of the tree is a transmitter that emits the football match, the leaves of the tree are the potential users and other vertices in the tree are relays (transmitters).

The price of transmission of a signal from one transmitter to another or to the user is given. A price of the entire broadcast is the **sum** of prices of all individual signal transmissions.

Every user is ready to pay a certain amount of money to watch the match and the TV-network then decides whether or not to provide the user with the signal.

Write a program that will find the **maximal** number of users able to watch the match so that the TV-network's **doesn't lose** money from broadcasting the match.

Input data

The first line of the input file contains two integers N and M, $2 \le N \le 3000$, $1 \le M \le N-1$, the number of vertices in the tree and the number of potential users.

The root of the tree is marked with the number 1, while other transmitters are numbered 2 to N-M and potential users are numbered N-M+1 to N.

The following N-M lines contain data about the transmitters in the following form:

$K\ A_1\ C_1\ A_2\ C_2\ \ldots\ A_K\ C_K$

Means that a transmitter transmits the signal to K transmitters or users, every one of them described by the pair of numbers A and C, the transmitter or user's number and the cost of transmitting the signal to them.

The last line contains the data about users, containing M integers representing respectively the price every one of them is willing to pay to watch the match.

Output data

The first and the only line of the output file should contain the maximal number of users described in the above text.

Examples

tele.in

```
5 3
2 2 2 5 3
2 3 2 4 3
3 4 2
```

tele.out

```
2
```

tele.in

```
5 3
2 2 2 5 3
2 3 2 4 3
4 4 2
```

tele.out

```
3
```

tele.in

```
9 6
3 2 2 3 2 9 3
2 4 2 5 2
3 6 2 7 2 8 2
4 3 3 3 1 1
```

tele.out

```
5
```

DOG

A park consists of paths forming a grid of 2,000,000 vertical and 2,000,000 horizontal paths with a distance of one meter between them. Paths are numbered 1 to 2,000,000 from top to bottom and left to right. There is a tree at the intersection of every two paths.

There are dogs numbered 1 to N playing in the park. Every dog has a chain of a certain length and a favourite tree. Dogs walk only on the paths and their chains follow them on the paths.

Every dog needs to be chained to some tree in the park so that:

- his favourite tree is in his **range**

- it can walk to every dog marked with a number **smaller** than its own, **wherever** that dog with the smaller number is.

Write a program that will find a tree for every dog to which it should be chained so that all the requests in the task description are satisfied.

Input data

The first line of the input file contains an integer N, 1 ≤ N ≤ 100,000, the number of dogs.

The following N lines contain data about the dogs, the (i+1)-th line contains data about the i-th dog in the following form:

R S D

This line means that the dog's favourite tree is located in the R-th row and S-th column and that the chain is D meters long. D is less than or equal to 100,000.

Output data

The output file consists of N lines. The i-th line contains the coordinates of the tree to which the i-th dog should be chained. The first coordinate is the row and the second is the column of the tree.

Note: the input data guarantees that a solution, while not necessarily unique, exists.

Examples

dog.in

```
2
6 8 2
11 11 5
```

dog.out

```
5 9
6 11
```

dog.in

```
3
11 11 2
9 11 2
10 18 4
```

dog.out

```
10 12
10 12
10 14
```

dog.in

```
6
21 27 1
23 27 3
19 27 5
21 33 6
23 29 6
26 30 7
```

dog.out

```
20 27
20 27
19 28
19 29
19 29
19 30
```