COMPETITION DAY 1 – OVERVIEW

TASK	ALIENS	FLOOD	SAILS
type	interactive	batch	batch
time limit (per test run)	1 second	2 seconds	1 second
memory limit (per test run)	64 MB	32 MB	64 MB
points	**100**	**100**	**100**
	300		

FLOOD

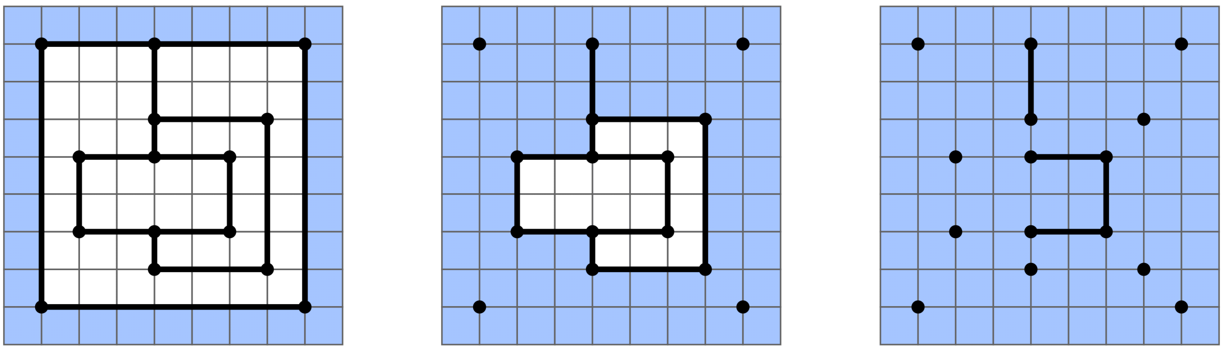
In 1964 a catastrophic flood struck the city of Zagreb. Many buildings were completely destroyed when the water struck their walls. In this task, you are given a simplified model of the city before the flood and you should determine which of the walls are left intact after the flood.

The model consists of N points in the coordinate plane and W walls. **Each wall connects a pair of points and does not go through any other points**. The model has the following additional properties:

- No two walls intersect or overlap, but they may touch at endpoints;
- Each wall is parallel to either the horizontal or the vertical coordinate axis.

Initially, the entire coordinate plane is dry. At time zero, water instantly floods the exterior (the space not bounded by walls). After exactly one hour, every wall with water on one side and air on the other breaks under the pressure of water. Water then floods the new area not bounded by any standing walls. Now, there may be new walls having water on one side and air on the other. After another hour, these walls also break down and water floods further. This procedure repeats until water has flooded the entire area.

An example of the process is shown in the following figure.



The state at time zero. Shaded cells represent the flooded area, while white cells represent dry area (air).

The state after one hour.

The state after two hours. Water has flooded the entire area and the 4 remaining walls cannot be broken down.

TASK

Write a program that, given the coordinates of the N points, and the descriptions of W walls connecting these points, determines which of the walls are left standing after the flood.

INPUT

The first line of input contains an integer N ($2 \le N \le 100\,000$), the number of points in the plane.

Each of the following N lines contains two integers X and Y (both between 0 and 1 000 000, inclusive), the coordinates of one point. The points are numbered 1 to N in the order in which they are given. No two points will be located at the same coordinates.

The following line contains an integer W ($1 \le W \le 2N$), the number of walls.

Each of the following W lines contains two different integers A and B ($1 \le A \le N$, $1 \le B \le N$), meaning that, before the flood, there was a wall connecting points A and B. The walls are numbered 1 to W in the order in which they are given.

OUTPUT

The first line of output should contain a single integer K, the number of walls left standing after the flood.

The following K lines should contain the indices of the walls that are still standing, one wall per line. The indices may be output in any order.

GRADING

In test cases worth a total of 40 points, all coordinates will be at most 500.

In those same cases, and cases worth another 15 points, the number of points will be at most 500.

DETAILED FEEDBACK WHEN SUBMITTING

During the contest, you may select up to 10 submissions for this task to be evaluated (as soon as possible) on part of the official test data. After the evaluation is done, a summary of the results will be available on the contest system.

EXAMPLE

input

```
15
1 1
8 1
4 2
7 2
2 3
4 3
6 3
2 5
4 5
6 5
4 6
7 6
1 8
4 8
8 8
17
1 2
2 15
15 14
14 13
13 1
14 11
11 12
12 4
4 3
3 6
6 5
5 8
8 9
9 11
9 10
10 7
7 6
```

output

```
4
6
15
16
17
```

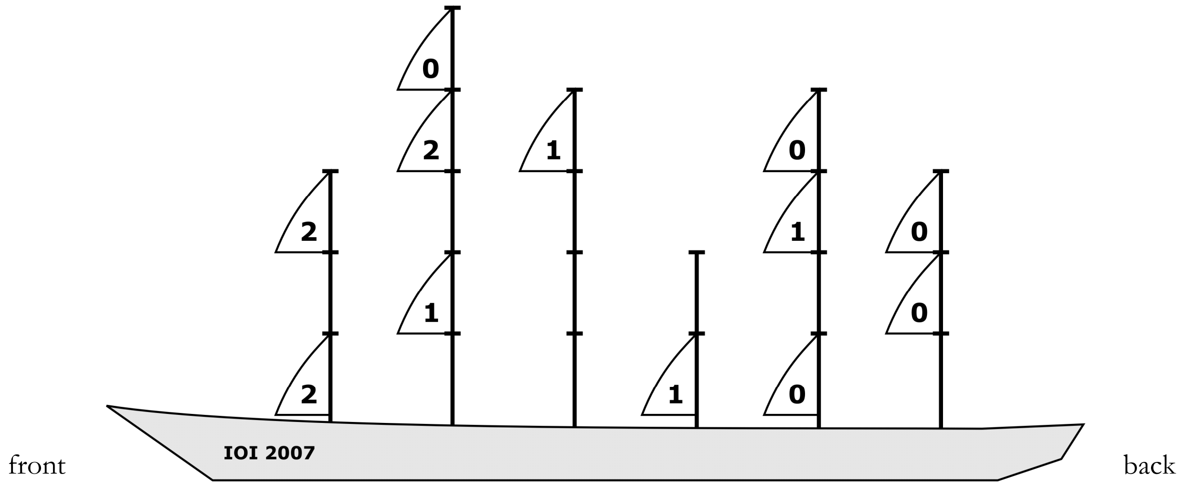
This example corresponds to the figure on the previous page.

SAILS

A new pirate sailing ship is being built. The ship has N masts (poles) divided into unit sized segments – the height of a mast is equal to the number of its segments. Each mast is fitted with a number of sails and each sail exactly fits into one segment. Sails on one mast can be arbitrarily distributed among different segments, but each segment can be fitted with at most one sail.

Different configurations of sails generate different amounts of thrust when exposed to the wind. Sails in front of other sails at the same height get less wind and contribute less thrust. For each sail we define its **inefficiency** as the total number of sails that are **behind** this sail and **at the same height**. Note that "in front of" and "behind" relate to the orientation of the ship: in the figure below, "in front of" means to the left, and "behind" means to the right.

The **total inefficiency** of a configuration is the sum of the inefficiencies of all individual sails.



This ship has 6 masts, of heights 3, 5, 4, 2, 4 and 3 from front (left side of image) to back.
This distribution of sails gives a total inefficiency of 10. The individual inefficiency of each sail is written inside the sail.

TASK

Write a program that, given the height and the number of sails on each of the N masts, determines the **smallest** possible total inefficiency.

INPUT

The first line of input contains an integer N ($2 \le N \le 100\,000$), the number of masts on the ship.

Each of the following N lines contains two integers H and K ($1 \le H \le 100\,000$, $1 \le K \le H$), the height and the number of sails on the corresponding mast. Masts are given in order from the front to the back of the ship.

OUTPUT

Output should consist of a single integer, the smallest possible total inefficiency.

Note: use a 64-bit integer type to calculate and output the result (`long long` in C/C++, `int64` in Pascal).

GRADING

In test cases worth a total of 25 points, the total number of ways to arrange the sails will be at most 1 000 000.

EXAMPLE

input

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

output

```
10
```

This example corresponds to the figure on the previous page.

MINERS

There are **two** coal mines, each employing a group of miners. Mining coal is hard work, so miners need food to keep at it. Every time a shipment of food arrives at their mine, the miners produce some amount of coal. There are three types of food shipments: meat shipments, fish shipments and bread shipments.

Miners like variety in their diet and they will be more productive if their food supply is kept varied. More precisely, every time a new shipment arrives to their mine, they will **consider the new shipment and the previous two shipments** (or fewer if there haven't been that many) and then:

- If all shipments were of the same type, they will produce one unit of coal.
- If there were two different types of food among the shipments, they will produce two units of coal.
- If there were three different types of food, they will produce three units of coal.

We know in advance the types of food shipments and the order in which they will be sent. It is possible to influence the amount of coal that is produced by determining which shipment should go to which mine. Shipments cannot be divided; each shipment must be sent to one mine or the other in its entirety.

The two mines don't necessarily have to receive the same number of shipments (in fact, it is permitted to send all shipments to one mine).

TASK

Your program will be given the types of food shipments, in the order in which they are to be sent. Write a program that finds the **largest total amount of coal** that can be produced (in both mines) by deciding which shipments should be sent to mine 1 and which shipments should be sent to mine 2.

INPUT

The first line of input contains an integer N ($1 \leq N \leq 100\,000$), the number of food shipments.

The second line contains a string consisting of N characters, the types of shipments in the order in which they are to be distributed. Each character will be one of the uppercase letters 'M' (for meat), 'F' (for fish) or 'B' (for bread).

OUTPUT

Output a single integer, the largest total amount of coal that can be produced.

GRADING

In test cases worth a total of 45 points, the number of shipments N will be at most 20.

DETAILED FEEDBACK WHEN SUBMITTING

During the contest, you may select up to 10 submissions for this task to be evaluated (as soon as possible) on part of the official test data. After the evaluation is done, a summary of the results will be available on the contest system.

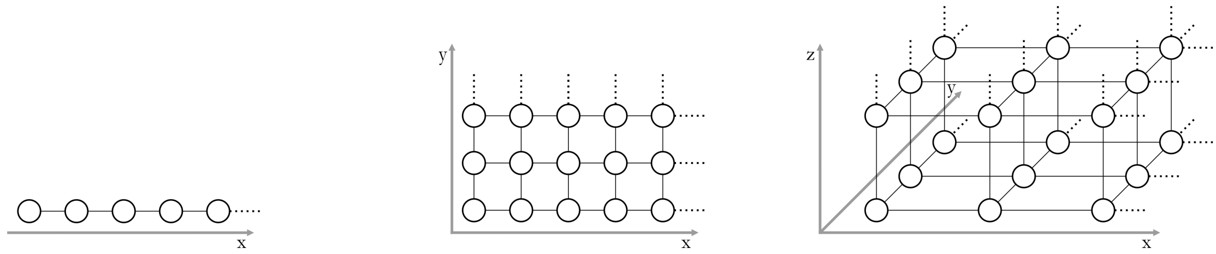
EXAMPLES

input	input
6 MBMFFB	16 MMBMBBBBMMMMMBMB
output	**output**
12	29

In the left sample, by distributing the shipments in this order: mine 1, mine 1, mine 2, mine 2, mine 1, mine 2, the shipments will result in 1, 2, 1, 2, 3 and 3 units of coal produced in that order, for a total of 12 units. There are other ways to achieve this largest amount.

PAIRS

Mirko and Slavko are playing with toy animals. First, they choose one of three boards given in the figure below. Each board consists of cells (shown as circles in the figure) arranged into a one, two or three dimensional grid.



Board 1 Board 2 Board 3

Mirko then places N little **toy animals** into the cells.

The **distance** between two cells is the smallest number of moves that an animal would need in order to reach one cell from the other. In one move, the animal may step into one of the adjacent cells (connected by line segments in the figure).

Two animals can hear each other if the distance between their cells is **at most D**. Slavko's task is to calculate how many **pairs of animals** there are such that one animal can hear the other.

TASK

Write a program that, given the board type, the locations of all animals, and the number D, finds the desired number of pairs.

INPUT

The first line of input contains four integers in this order:

- The board type B ($1 \leq B \leq 3$);
- The number of animals N ($1 \leq N \leq 100\,000$);
- The largest distance D at which two animals can hear each other ($1 \leq D \leq 100\,000\,000$);
- The size of the board M (the largest coordinate allowed to appear in the input):
  - When B=1, M will be at most 75 000 000.
  - When B=2, M will be at most 75 000.
  - When B=3, M will be at most 75.

Each of the following N lines contains B integers separated by single spaces, the coordinates of one toy animal. Each coordinate will be between 1 and M (inclusive).

More than one animal may occupy the same cell.

OUTPUT

Output should consist of a single integer, the number of pairs of animals that can hear each other.

Note: use a 64-bit integer type to calculate and output the result (`long long` in C/C++, `int64` in Pascal).

GRADING

In test cases worth a total of 30 points, the number of animals N will be at most 1 000.

Furthermore, for each of the three board types, a solution that correctly solves all test cases of that type will be awarded at least 30 points.

EXAMPLES

input

```
1 6 5 100
25
50
50
10
20
23
```

output

```
4
```

input

```
2 5 4 10
5 2
7 2
8 4
6 5
4 4
```

output

```
8
```

input

```
3 8 10 20
10 10 10
10 10 20
10 20 10
10 20 20
20 10 10
20 10 20
20 20 10
20 20 20
```

output

```
12
```

Clarification for the leftmost example. Suppose the animals are numbered 1 through 6 in the order in which they are given. The four pairs are:

- 1-5 (distance 5)
- 1-6 (distance 2)
- 2-3 (distance 0)
- 5-6 (distance 3)

Clarification for the middle example. The eight pairs are:

- 1-2 (distance 2)
- 1-4 (distance 4)
- 1-5 (distance 3)
- 2-3 (distance 3)
- 2-4 (distance 4)
- 3-4 (distance 3)
- 3-5 (distance 4)
- 4-5 (distance 3)

TRAINING

Mirko and Slavko are training hard for the annual tandem cycling marathon taking place in Croatia. They need to choose a route to train on.

There are N cities and M roads in their country. Every road connects two cities and can be traversed in both directions. Exactly N−1 of those roads are **paved**, while the rest of the roads are unpaved trails. Fortunately, the network of roads was designed so that each pair of cities is connected by a path consisting of paved roads. In other words, the N cities and the N−1 **paved roads form a tree structure**.

Additionally, each city is an endpoint for **at most 10 roads total**.

A training route starts in some city, follows some roads and ends in the same city it started in. Mirko and Slavko like to see new places, so they made a rule **never to go through the same city** nor **travel the same road twice**. The training route may start in any city and does not need to visit every city.

Riding in the back seat is easier, since the rider is shielded from the wind by the rider in the front. Because of this, Mirko and Slavko change seats in every city. To ensure that they get the same amount of training, they must choose a route with an **even number of roads**.

Mirko and Slavko's competitors decided to **block** some of the unpaved roads, making it **impossible** for them to find a training route satisfying the above requirements. For each unpaved road there is a **cost** (a **positive** integer) associated with blocking the road. It is impossible to block paved roads.

TASK

Write a program that, given the description of the network of cities and roads, finds the **smallest total cost** needed to block the roads so that **no training route exists** satisfying the above requirements.

INPUT

The first line of input contains two integers N and M ($2 \le N \le 1\,000$, $N-1 \le M \le 5\,000$), the number of cities and the total number of roads.

Each of the following M lines contains three integers A, B and C ($1 \le A \le N$, $1 \le B \le N$, $0 \le C \le 10\,000$), describing one road. The numbers A and B are different and they represent the cities directly connected by the road. If C=0, the road is paved; otherwise, the road is unpaved and C represents the cost of blocking it.

Each city is an endpoint for at most 10 roads. There will never be more than one road directly connecting a single pair of cities.

OUTPUT

Output should consist of a single integer, the smallest total cost as described in the problem statement.

GRADING

In test cases worth a total of 30 points, the paved roads will form a chain (that is, no city will be an endpoint for three or more paved roads).

DETAILED FEEDBACK WHEN SUBMITTING

During the contest, you may select up to 10 submissions for this task to be evaluated (as soon as possible) on part of the official test data. After the evaluation is done, a summary of the results will be available on the contest system.

EXAMPLES

input

```
5 8
2 1 0
3 2 0
4 3 0
5 4 0
1 3 2
3 5 2
2 4 5
2 5 1
```

output

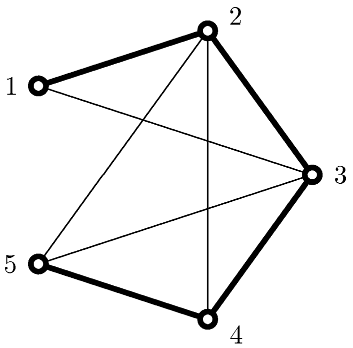
```
5
```

input

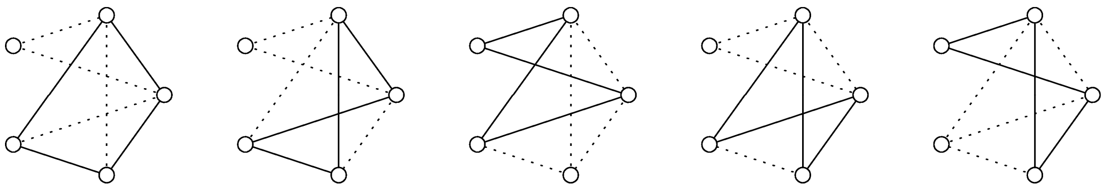
```
9 14
1 2 0
1 3 0
2 3 14
2 6 15
3 4 0
3 5 0
3 6 12
3 7 13
4 6 10
5 6 0
5 7 0
5 8 0
6 9 11
8 9 0
```

output

```
48
```



The layout of the roads and cities in the first example. Paved roads are shown in bold.



There are five possible routes for Mirko and Slavko. If the roads 1-3, 3-5 and 2-5 are blocked, then Mirko and Slavko cannot use any of the five routes. The cost of blocking these three roads is 5.

It is also possible to block just two roads, 2-4 and 2-5, but this would result in a higher cost of 6.