Tinkoff Challenge - Elimination Round

A. Oleg and shares

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Oleg the bank client checks share prices every day. There are n share prices he is interested in. Today he observed that each second exactly one of these prices decreases by k rubles (note that each second exactly one price changes, but at different seconds different prices can change). Prices can become negative. Oleg found this process interesting, and he asked Igor the financial analyst, what is the minimum time needed for all n prices to become equal, or it is impossible at all? Igor is busy right now, so he asked you to help Oleg. Can you answer this question?

Input

The first line contains two integers n and k $(1 \le n \le 10^5, 1 \le k \le 10^9)$ — the number of share prices, and the amount of rubles some price decreases each second.

The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^9)$ — the initial prices.

Output

Print the only line containing the minimum number of seconds needed for prices to become equal, of «-1» if it is impossible.

Examples

input
```
3 3
12 9 15
```
output
```
3
```

input
```
2 2
10 9
```
output
```
-1
```

input
```
4 1
1 1000000000 1000000000 1000000000
```
output
```
2999999997
```

Note

Consider the first example.

Suppose the third price decreases in the first second and become equal 12 rubles, then the first price decreases and becomes equal 9 rubles, and in the third second the third price decreases again and becomes equal 9 rubles. In this case all prices become equal 9 rubles in 3 seconds.

There could be other possibilities, but this minimizes the time needed for all prices to become equal. Thus the answer is 3.

In the second example we can notice that parity of first and second price is different and never changes within described process. Thus prices never can become equal.

In the third example following scenario can take place: firstly, the second price drops, then the third price, and then fourth price. It happens 999999999 times, and, since in one second only one price can drop, the whole process takes $999999999 * 3 = 2999999997$ seconds. We can note that this is the minimum possible time.

B. Igor and his way to work

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Woken up by the alarm clock Igor the financial analyst hurried up to the work. He ate his breakfast and sat in his car. Sadly, when he opened his GPS navigator, he found that some of the roads in Bankopolis, the city where he lives, are closed due to road works. Moreover, Igor has some problems with the steering wheel, so he can make **no more than two turns** on his way to his office in bank.

Bankopolis looks like a grid of n rows and m columns. Igor should find a way from his home to the bank that has no more than two turns and doesn't contain cells with road works, or determine that it is impossible and he should work from home. A turn is a change in movement direction. Igor's car can only move to the left, to the right, upwards and downwards. Initially Igor can choose any direction. Igor is still sleepy, so you should help him.

Input

The first line contains two integers n and m $(1 \le n, m \le 1000)$ — the number of rows and the number of columns in the grid.

Each of the next n lines contains m characters denoting the corresponding row of the grid. The following characters can occur:

- "`.`" — an empty cell;
- "`*`" — a cell with road works;
- "`S`" — the cell where Igor's home is located;
- "`T`" — the cell where Igor's office is located.

It is guaranteed that "`S`" and "`T`" appear exactly once each.

Output

In the only line print "`YES`" if there is a path between Igor's home and Igor's office with no more than two turns, and "`NO`" otherwise.

Examples

input

```
5 5
..S..
****.
T....
****.
.....
```

output

```
YES
```

input

```
5 5
S....
****.
.....
.****
..T..
```

output

```
NO
```

Note

The first sample is shown on the following picture:

In the second sample it is impossible to reach Igor's office using less that 4 turns, thus there exists no path using no more than 2 turns. The path using exactly 4 turns is shown on this picture:

C. Mice problem

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Igor the analyst fell asleep on the work and had a strange dream. In the dream his desk was crowded with computer mice, so he bought a mousetrap to catch them.

The desk can be considered as an infinite plane, then the mousetrap is a rectangle which sides are parallel to the axes, and which opposite sides are located in points (x_1, y_1) and (x_2, y_2).

Igor wants to catch all mice. Igor has analysed their behavior and discovered that each mouse is moving along a straight line with constant speed, the speed of the i-th mouse is equal to (v_i^x, v_i^y), that means that the x coordinate of the mouse increases by v_i^x units per second, while the y coordinates increases by v_i^y units. The mousetrap is open initially so that the mice are able to move freely on the desk. Igor can close the mousetrap at any moment catching all the mice that are **strictly** inside the mousetrap.

Igor works a lot, so he is busy in the dream as well, and he asks you to write a program that by given mousetrap's coordinates, the initial coordinates of the mice and their speeds determines the earliest time moment in which he is able to catch all the mice. Please note that Igor can close the mousetrap only once.

Input

The first line contains single integer n $(1 \le n \le 100\,000)$ — the number of computer mice on the desk.

The second line contains four integers x_1, y_1, x_2 and y_2 $(0 \le x_1 \le x_2 \le 100\,000)$, $(0 \le y_1 \le y_2 \le 100\,000)$ — the coordinates of the opposite corners of the mousetrap.

The next n lines contain the information about mice.

The i-th of these lines contains four integers r_i^x, r_i^y, v_i^x and v_i^y, $(0 \le r_i^x, r_i^y \le 100\,000,\ -100\,000 \le v_i^x, v_i^y \le 100\,000)$, where (r_i^x, r_i^y) is the initial position of the mouse, and (v_i^x, v_i^y) is its speed.

Output

In the only line print minimum possible non-negative number t such that if Igor closes the mousetrap at t seconds from the beginning, then all the mice are **strictly** inside the mousetrap. If there is no such t, print `-1`.

Your answer is considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is considered correct if .

Examples

input

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

output

```
0.57142857142857139685
```

input

```
4
7 7 9 8
0 3 -5 4
5 0 5 4
9 9 -1 -6
10 5 -7 -10
```

output

```
-1
```

Note

Here is a picture of the first sample

Points A, B, C, D - start mice positions, segments are their paths.

Then, at first time when all mice will be in rectangle it will be looks like this:

Here is a picture of the second sample

Points A, D, B will never enter rectangle.

D. Presents in Bankopolis

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bankopolis is an incredible city in which all the n crossroads are located on a straight line and numbered from 1 to n along it. On each crossroad there is a bank office.

The crossroads are connected with m oriented bicycle lanes (the i-th lane goes from crossroad u_i to crossroad v_i), the difficulty of each of the lanes is known.

Oleg the bank client wants to gift happiness and joy to the bank employees. He wants to visit exactly k offices, in each of them he wants to gift presents to the employees.

The problem is that Oleg don't want to see the reaction on his gifts, so he can't use a bicycle lane which passes near the office in which he has already presented his gifts (formally, the i-th lane passes near the office on the x-th crossroad if and only if $min(u_i, v_i) < x < max(u_i, v_i)$)). Of course, in each of the offices Oleg can present gifts exactly once. Oleg is going to use exactly $k - 1$ bicycle lane to move between offices. Oleg can start his path from any office and finish it in any office.

Oleg wants to choose such a path among possible ones that the total difficulty of the lanes he will use is minimum possible. Find this minimum possible total difficulty.

Input

The first line contains two integers n and k $(1 \le n, k \le 80)$ — the number of crossroads (and offices) and the number of offices Oleg wants to visit.

The second line contains single integer m $(0 \le m \le 2000)$ — the number of bicycle lanes in Bankopolis.

The next m lines contain information about the lanes.

The i-th of these lines contains three integers u_i, v_i and c_i $(1 \le u_i, v_i \le n,\ 1 \le c_i \le 1000)$, denoting the crossroads connected by the i-th road and its difficulty.

Output

In the only line print the minimum possible total difficulty of the lanes in a valid path, or `-1` if there are no valid paths.

Examples

input

```
7 4
4
1 6 2
6 2 2
2 4 2
2 7 1
```

output

```
6
```

input

```
4 3
4
2 1 2
1 3 2
3 4 2
4 1 1
```

output

```
3
```

Note

In the first example Oleg visiting banks by path $1 \to 6 \to 2 \to 4$.

Path $1 \to 6 \to 2 \to 7$ with smaller difficulty is incorrect because crossroad $2 \to 7$ passes near already visited office on the crossroad 6.

In the second example Oleg can visit banks by path $4 \to 1 \to 3$.

E. Problem of offices

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Earlier, when there was no Internet, each bank had a lot of offices all around Bankopolis, and it caused a lot of problems. Namely, each day the bank had to collect cash from all the offices.

Once Oleg the bank client heard a dialogue of two cash collectors. Each day they traveled through all the departments and offices of the bank following the same route every day. The collectors started from the central department and moved between some departments or between some department and some office using special roads. Finally, they returned to the central department. The total number of departments and offices was n, the total number of roads was $n - 1$. In other words, the special roads system was a rooted tree in which the root was the central department, the leaves were offices, the internal vertices were departments. The collectors always followed the same route in which the number of roads was minimum possible, that is $2n - 2$.

One of the collectors said that the number of offices they visited between their visits to offices a and then b (in the given order) is equal to the number of offices they visited between their visits to offices b and then a (in this order). The other collector said that the number of offices they visited between their visits to offices c and then d (in this order) is equal to the number of offices they visited between their visits to offices d and then c (in this order). The interesting part in this talk was that the shortest path (using special roads only) between any pair of offices among a, b, c and d **passed through the central department**.

Given the special roads map and the indexes of offices a, b, c and d, determine if the situation described by the collectors was possible, or not.

Input

The first line contains single integer n ($5 \le n \le 5000$) — the total number of offices and departments. The departments and offices are numbered from 1 to n, the central office has index 1.

The second line contains four integers a, b, c and d ($2 \le a, b, c, d \le n$) — the indexes of the departments mentioned in collector's dialogue. It is guaranteed that these indexes are offices (i.e. leaves of the tree), not departments. It is guaranteed that the shortest path between any pair of these offices passes through the central department.

On the third line $n - 1$ integers follow: $p_2, p_3, ..., p_n$ ($1 \le p_i < i$), where p_i denotes that there is a special road between the i-th office or department and the p_i-th department.

Please note the joint enumeration of departments and offices.

It is guaranteed that the given graph is a tree. The offices are the leaves, the departments are the internal vertices.

Output

If the situation described by the cash collectors was possible, print "YES". Otherwise, print "NO".

Examples

input

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

output

```
YES
```

input

```
10
3 8 9 10
1 2 2 2 2 2 1 1 1
```

output

```
NO
```

input

```
13
13 12 9 7
1 1 1 1 5 5 2 2 2 3 3 4
```

output

```
YES
```

Note

In the first example the following collector's route was possible: . We can note that between their visits to offices a and b the collectors visited the same number of offices as between visits to offices b and a; the same holds for c and d (the collectors' route is infinite as they follow it each day).

In the second example there is no route such that between their visits to offices c and d the collectors visited the same number of offices as between visits to offices d and c. Thus, there situation is impossible.

In the third example one of the following routes is: .

F. Julia the snail

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

After hard work Igor decided to have some rest.

He decided to have a snail. He bought an aquarium with a slippery tree trunk in the center, and put a snail named Julia into the aquarium.

Igor noticed that sometimes Julia wants to climb onto the trunk, but can't do it because the trunk is too slippery. To help the snail Igor put some ropes on the tree, fixing the lower end of the i-th rope on the trunk on the height l_i above the ground, and the higher end on the height r_i above the ground.

For some reason no two ropes share the same position of the higher end, i.e. all r_i are distinct. Now Julia can move down at any place of the trunk, and also move up from the lower end of some rope to its higher end. Igor is proud of his work, and sometimes think about possible movements of the snail. Namely, he is interested in the following questions: «Suppose the snail is on the trunk at height x now. What is the highest position on the trunk the snail can get on if it would never be lower than x or higher than y?» Please note that Julia can't move from a rope to the trunk before it reaches the higher end of the rope, and Igor is interested in the highest position **on the tree trunk**.

Igor is interested in many questions, and not always can answer them. Help him, write a program that answers these questions.

Input

The first line contains single integer n ($1 \le n \le 100000$) — the height of the trunk.

The second line contains single integer m ($1 \le m \le 100000$) — the number of ropes.

The next m lines contain information about the ropes.

The i-th of these lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the heights on which the lower and the higher ends of the i-th rope are fixed, respectively. It is guaranteed that all r_i are distinct.

The next line contains single integer q ($1 \le q \le 100000$) — the number of questions.

The next q lines contain information about the questions.

Each of these lines contain two integers x and y ($1 \le x \le y \le n$), where x is the height where Julia starts (and the height Julia can't get lower than), and y is the height Julia can't get higher than.

Output

For each question print the maximum reachable for Julia height.

Examples

input

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

output

```
2
2
5
5
7
```

input

```
10
10
3 7
1 4
1 6
5 5
1 1
3 9
7 8
1 2
3 3
7 10
10
2 4
```

```
1 7
3 4
3 5
2 8
2 5
5 5
3 5
7 7
3 10
```

output

```
2
7
3
3
2
2
5
3
7
10
```

Note

The picture of the first sample is on the left, the picture of the second sample is on the right. Ropes' colors are just for clarity, they don't mean anything.

G. Oleg and chess

time limit per test: 6.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Oleg the bank client solves an interesting chess problem: place on $n \times n$ chessboard the maximum number of rooks so that they don't beat each other. Of course, no two rooks can share the same cell.

Remind that a rook standing in the cell (a, b) beats a rook standing in the cell (x, y) if and only if $a = x$ or $b = y$.

Unfortunately (of fortunately?) for Oleg the answer in this problem was always n, so the task bored Oleg soon. He decided to make it more difficult by removing some cells from the board. If a cell is deleted, Oleg can't put a rook there, but rooks do beat each other "through" deleted cells.

Oleg deletes the cells in groups, namely, he repeatedly choose a rectangle with sides parallel to the board sides and deletes all the cells inside the rectangle. Formally, if he chooses a rectangle, lower left cell of which has coordinates (x_1, y_1), and upper right cell of which has coordinates (x_2, y_2), then he deletes all such cells with coordinates (x, y) that $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$. It is guaranteed that no cell is deleted twice, i.e. the chosen rectangles do not intersect.

This version of the problem Oleg can't solve, and his friend Igor is busy at a conference, so he can't help Oleg.

You are the last hope for Oleg! Help him: given the size of the board and the deleted rectangles find the maximum possible number of rooks that could be placed on the board so that no two rooks beat each other.

Input

The first line contains single integer n ($1 \le n \le 10000$) — the size of the board.

The second line contains single integer q ($0 \le q \le 10000$) — the number of deleted rectangles.

The next q lines contain the information about the deleted rectangles.

Each of these lines contains four integers x_1, y_1, x_2 and y_2 ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le n$) — the coordinates of the lower left and the upper right cells of a deleted rectangle.

If is guaranteed that the rectangles do not intersect.

Output

In the only line print the maximum number of rooks Oleg can place on the board so that no two rooks beat each other.

Examples

input

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

output

```
3
```

input

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

output

```
8
```

Note

Here is the board and the example of rooks placement in the first example: