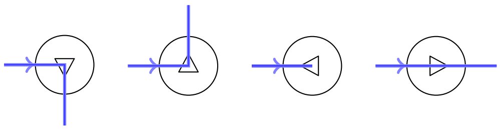
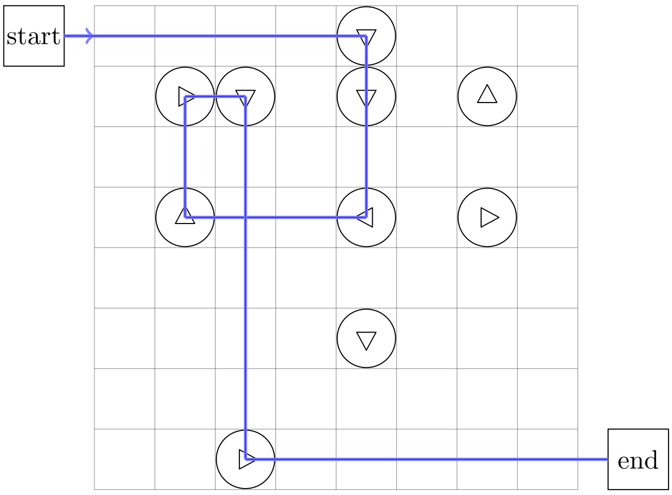
Problem A - Lasers
UAC4 - 2015

You are given an n by m grid that contains k mirrors. Each mirror can be set into four different positions. When a mirror is set in the first position it will send all incoming light down, in the second position it will send all the light up, in the third position it will send the light left and in the forth position it will send the light to the right. The following figure shows the four positions and the effect on the light coming from the left. The direction to which the light is reflected depends only on the position of the mirror, not on the direction it comes from.



A light emissor is set to the left of the top left corner of the grid and a receptor is set to the right of the bottom right corner of the grid.

Your task is to, given the positions of the mirrors, find if there exists a configuration that allows the light to travel from the emissor to the receptor. The following image shows such a configuration. Note that the position of some mirrors is irrelevant in this case.



Note that the path of the light can have self-intersections.

Input

The first line of the input contains three integers, n, m and k as described above. Then follow k lines, one for each mirror, each with two integers r and c giving

the row and the column of the mirror, respectively.
The top left corner of the grid is position $(0, 0)$ and the bottom right corner is position $(n-1, m-1)$.

Constraints

- $1 \leq n, m \leq 10^6$
- $0 \leq k \leq \min(10^4, n \cdot m)$
- No two mirrors are at the same position.

Output

A single line with `yes` if there exists a configuration of the mirrors that allows the light to travel from the emissor to the receptor and `no` otherwise.

Sample Test Cases

Sample Input 1

```
9 9 8
0 4
4 4
4 1
6 1
6 7
7 0
7 7
8 0
```

Sample Output 1

```
yes
```

Sample Input 2

```
9 12 14
0 1
0 11
1 6
1 9
2 6
2 11
3 0
3 1
4 7
```

```
4 9
5 8
6 0
7 0
8 8
```

Sample Output 2

```
no
```

Problem B - Watching a race
UAC4 - 2015

There is a group of n competitors racing. You are watching the race on the TV but you get distracted at some point at miss a part of the race. After a while you continue watching the race. You would like to know if you missed a lot of interesting things. Since you remember the standings before you got distracted, you would like to know, given the current standings, what is the minimum number of overtakes that took place while you where distacred.

Example:

Suppose that before you go distracted the standings where 1 2 3 4 5 so that competitor 1 is in firth place, competitor 2 is in second place and so on. If after you got distracted the standings are 3 1 2 5 4 then competitors 1 and 2 must have been overtaken by competitor 3 and competitor 4 must have been overtaked by competitor 5. Thus the minimum number of overtakes that took place is equal to 3.

Input

The first line contains a single integer n giving the number of competitors. Then follow two lines, one with the standings before you got distracted and the other with the standings after you started watching the race again. Each of those lines contain n integers separated by single spaces. The i-th element on a line is the id of the competitor that is at the i-th position. Competitors are numbered from 1 to n.

Constraints

- $2 \leq n \leq 100$

Output

A single integer with the minimum number of overtakes that took place.

Sample Test Cases

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 2

```
4
```

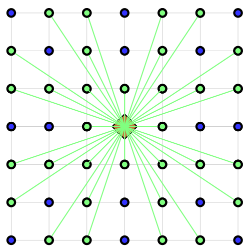
Problem C - Forest in grid-land
UAC4 - 2015

In grid-land there is a very big forest. The forest contains a tree in each integer coordinate point (x, y) such that $-n \leq x, y \leq n$ except position $(0, 0)$. To monitor fires, the villagers want to install a watch tower at position $(0, 0)$. From that tower, they can watch over all trees such that the line that connects the watch tower to the tree does not pass through any other tree.

Given n, your task is to compute how many trees can the be seen from the watch tower?

Example:

In the following picture $n = 3$, the watch tower (in the center) can see the 32 trees in green.



Input

A single line containing the size of the forest, n, as described above.

Constraints

- $1 \leq n \leq 10^8$

Output

A single line with an integer giving the number of trees that the watch tower. The answer is at most $2^{63} - 1$.

Sample Test Cases

Sample Input 1

```
2
```

Sample Output 1

```
16
```

Sample Input 2

```
3
```

Sample Output 2

```
32
```

Problem D - Renaming folders
UAC4 - 2015

You are preparing a programming contest with $n \leq 26$ problems. Your problems are named with uppercase character `A`, `B`, etc. They are all in the same folder and they are in increasing order of difficulty, so that, the easiest problem is `A`, the second easiest is `B` and so on. You want to rename them to change the order in which they will appear. The problem is that at any given time, no two folders can have the same name. Formally a renaming is perfomed as follows:

1. choose a problem p
2. choose any string s that is different from the name of any problem
3. change the name of p to s

What is the minimum number of rename operations necessary to achieve a given permutation of the names?

Example:

Suppose you have 5 problems and you want to rename them to:

$$\begin{array}{rccccc}
\text{initial:} & \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\text{target:} & \texttt{B} & \texttt{D} & \texttt{A} & \texttt{E} & \texttt{C}
\end{array}$$

One way to do this is to first rename each problem with a temporary name and then rename each of the with the desiered name:

$$\begin{array}{rcccccl}
\text{initial:} & \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{temporary:} & \texttt{T1} & \texttt{T2} & \texttt{T3} & \texttt{T4} & \texttt{T5} & \text{(costs 5 renames)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{final:} & \texttt{B} & \texttt{D} & \texttt{A} & \texttt{E} & \texttt{C} & \text{(costs 5 renames)}
\end{array}$$

This gives a total of 10 renames. We can do with only 6 rename operations:

$$\begin{array}{rcccccl}
\text{initial:} & \texttt{A} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 1:} & \texttt{T} & \texttt{B} & \texttt{C} & \texttt{D} & \texttt{E} & \text{(costs 1 rename)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 2:} & \texttt{T} & \texttt{B} & \texttt{A} & \texttt{D} & \texttt{E} & \text{(costs 1 rename)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 3:} & \texttt{T} & \texttt{B} & \texttt{A} & \texttt{D} & \texttt{C} & \text{(costs 1 rename)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 3:} & \texttt{T} & \texttt{B} & \texttt{A} & \texttt{E} & \texttt{C} & \text{(costs 1 rename)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 4:} & \texttt{T} & \texttt{D} & \texttt{A} & \texttt{E} & \texttt{C} & \text{(costs 1 rename)} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{step 5:} & \texttt{B} & \texttt{D} & \texttt{A} & \texttt{E} & \texttt{C} & \text{(costs 1 rename)}
\end{array}$$

Input

The first line contains a single integer n giving the number of problems. Then follows a line giving the target names of the problems. The i-th character of the line denote the target name of the i-th problem. Initially the first problem is named `A`, the second `B` and so on.

Constraints

- $1 \leq n \leq 26$

Output

A single line with the minimum number of operations needed to rename the problems.

Sample Test Cases

Sample Input 1

```
1
A
```

Sample Output 1

```
0
```

Sample Input 2

```
5
B D A E C
```

Sample Output 2

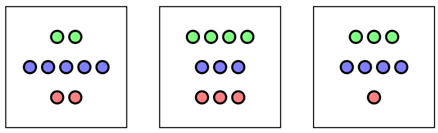
```
6
```

Problem E - Sorting the marbles
UAC4 - 2015

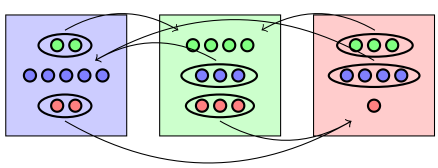
Little Jack loves to collect marbles. He has a lot marbles of n different colors. However he is not very organized and has them spread into n boxes. Each box contains some marbles of one or more colors. He decided to rearrange the marbles so that each box contains only marbles of the same color. For that, he will assign a different color to each box and then move all the marbles to the box of the corresponding color. To minimize the time this will take, he wants to find which colors should be assigned to each of the boxes as to minimize the total number of marbles that need to be moved.

Example:

Suppose initially the boxes are filled as shown in the next picture.



If he labels the first box as the blue box, the second as the green box and the thirds as the red box, then the total cost is $5+7+5=17$. The following picture shows the movements. This is not optimal. If instead he labels the first box as blue, then second as red and the third as green the total cost is $7+3+6=16$.



Input

The first line of the input contains one integer n, the total number of boxes (and colors). The follow n lines each with n integers. The element j-th element on line i, c_{ij}, represents the number of balls of color i initially in box j.

Constraints

- $2 \leq n \leq 50$
- $1 \leq c_{ij} \leq 100$

Output

A single line with an integers representing the minimum number of balls that have to be moved in an optimal labeling of the boxes.

Sample Test Cases

Sample Input 1

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

Sample Output 1

```
16
```

The first input corresponds to the example above.

Sample Input 2

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

Sample Output 2

```
22
```