Love Polygon

Problem ID: polygon

Time limit: 2 seconds

As we all know, TV soap operas with many characters can lead to seriously complicated love dramas. In one TV show, there are N characters. Each character loves exactly one character (which could actually be him- or herself). We say that two characters are in a relationship if and only if they love each other.

One particular type of complication is called a "love polygon". We say that 3 or more characters are in a "love polygon" if the first character loves the second, the second loves the third and so on, and also the last character loves the first.

Recent polling has shown that viewers have grown tired of this drama and would prefer something more romantic. Therefore, it was decided to shoot some characters with love arrows so that everyone is in a relationship. By shooting someone with a love arrow, you can change whom that character loves (to any character of your choice).

What is the least number of love arrows needed to get everyone into a relationship?

Input

The first line of the input contains the integer N, the number of characters involved. The next N lines all contain two space-separated names s and t, meaning that the character named s initially loves the character named t. Names of the characters are no more than 10 letters long and consist of lowercase English letters.

Output

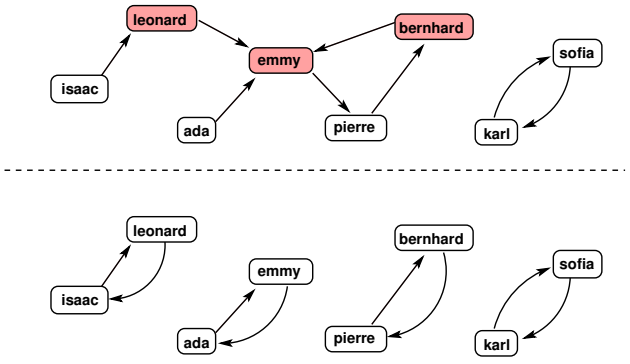
Output one integer – the minimum number of love arrows needed to get everyone into a relationship. If this is not possible, output `-1`.

Constraints

Your solution will be tested on a set of test groups, each worth a number of points. Each test group contains a set of test cases. To get the points for a test group you need to solve all test cases in the test group. Your final score will be the maximum score of a single submission.

Group	Points	Limits	Additional Constraints
1	21	$2 \leq N \leq 20$	
2	25	$2 \leq N \leq 100\,000$	Each character is loved by someone (possibly themselves).
3	29	$2 \leq N \leq 100\,000$	There are no relationships or "love polygons".
4	25	$2 \leq N \leq 100\,000$	

Explanation of Samples



The first example is illustrated in the figure above. The upper part shows the initial love situation, where an arrow pointing from s to t indicates that s initially loves t, and the pink color highlights the three characters that needs to be shot with love arrows in the unique optimal solution. The lower part shows the situation afterwards.

In the second example (which satisfies the constraints of group 3), there are several optimal solutions. One of them is to shoot `a`, `b` and `d` with love arrows, and have them fall in love with `b`, `a` and `c`, respectively.

In the third example, we have a love triangle, where no matter how many love arrows we shoot, someone will always be left out.

Sample Input 1

```
8
leonard emmy
ada emmy
isaac leonard
emmy pierre
pierre bernhard
bernhard emmy
sofia karl
karl sofia
```

Sample Output 1

```
3
```

Sample Input 2

```
4
a c
b c
c d
d d
```

Sample Output 2

```
3
```

Sample Input 3

```
3
rocky scarlet
scarlet patrick
patrick rocky
```

Sample Output 3

```
-1
```

Martian DNA

Problem ID: dna

Time limit: 2 seconds

As you are probably aware, human DNA can be represented as a long string over an alphabet of size four (A, C, G, T), where each symbol represents a distinct nucleobase (respectively; adenine, cytosine, guanine, and thymine).

For martians, however, things are a bit different; research conducted on the latest martian captured by NASA revealed that martian DNA consists of a whopping K distinct nucleobases! Martian DNA may thus be represented as a string over an alphabet of size K.

Now, a certain research group interested in exploiting martian DNA in artificial intelligence applications has requested to get a single consecutive piece of a martian DNA string. For R of the nucleobases, they have specified a minimum quantity of how many they need of that particular nucleobase to be present in their sample.

You are interested in finding the shortest substring of the DNA which satisfies their requirements.

Input

The first line contains three integers N, K, and R, denoting the total length of the martian DNA, the alphabet size, and the number of nucleobases for which the researchers have a minimum quantity requirement, respectively. They obey $1 \leq R \leq K \leq N$.

The second line contains N space-separated integers: the complete martian DNA string. The i-th of these integers, D_i, indicates what nucleobase is in the i-th position of the DNA string. Nucleobases are 0-indexed, i.e. $0 \leq D_i < K$. Each nucleobase will occur at least once in the DNA string.

Each of the following R lines contains two integers B and Q representing a nucleobase and its mimimum required quantity, respectively ($0 \leq B < K, 1 \leq Q \leq N$). No nucleobase will be listed more than once in these R lines.

Output

Output a single integer, the length of the shortest consecutive substring of the DNA that satisfies the researchers' requirements. If no such substring exists, output "`impossible`".

Constraints

Your solution will be tested on a set of test groups, each worth a number of points. Each test group contains a set of test cases. To get the points for a test group you need to solve all test cases in the test group. Your final score will be the maximum score of a single submission.

Group	Points	Limits
1	16	$1 \leq N \leq 100, R \leq 10$
2	24	$1 \leq N \leq 4\,000, R \leq 10$
3	28	$1 \leq N \leq 200\,000, R \leq 10$
4	32	$1 \leq N \leq 200\,000$

Explanation of Samples

In the first sample, there are three substrings of length 2 that contain one each of nucleobases 0 and 1 (namely "`0 1`", "`1 0`" and "`0 1`"), but no substrings of length 1. Thus the shortest length is 2.

In the second sample, the (unique) optimal substring is "`1 3 2 0 1 2 0`".

In the third sample, there aren't enough nucleobases of type 0.

Sample Input 1	Sample Output 1
5 2 2 0 1 1 0 1 0 1 1 1	2

Sample Input 2

```
13 4 3
1 1 3 2 0 1 2 0 0 0 0 3 1
0 2
2 1
1 2
```

Sample Output 2

```
7
```

Sample Input 3

```
5 3 1
1 2 0 1 2
0 2
```

Sample Output 3

```
impossible
```

Worm Worries
Problem ID: worm
Time limit: 3 seconds

You are looking for a place in the soil to put your pet worm, Maximus. You limit your search to a box-shaped region with dimensions $N \times M \times K$ centimeters which you have divided into a three-dimensional grid of cube-centimeter cells, labeled (x, y, z) after their position in the grid ($1 \leq x \leq N$, $1 \leq y \leq M$, $1 \leq z \leq K$). Each cell has a certain humidity $H[x, y, z]$ which is an integer in the range $1 \ldots 10^9$. You can measure the humidity of any cell with a special sensor.

Maximus loves humid places, so you need to put him in a cell which is at least as humid as its neighboring cells, otherwise he goes away and you will have trouble finding him. In other words, you need to place Maximus in a local maximum. More precisely, you need to find a cell (x, y, z), such that

$$H[x, y, z] \geq \max(H[x+1, y, z], H[x-1, y, z], H[x, y+1, z], H[x, y-1, z], H[x, y, z+1], H[x, y, z-1]),$$

where a value is treated as 0 when it is outside the box (because Maximus absolutely wants to stay in the box).

However, the number of cells can be very large, so you do not want to measure the humidity of all the cells. Therefore, in this task, you are allowed to interact with the grader, and ask for the humidity at given points. When you have found a suitable location for Maximus, give that location to the grader.

Interactivity

The first line of the input contains four positive integers: N, M, K and Q, where N, M and K are the box dimensions and Q is the maximum number of measurements you may perform.

After that, you can write at most Q lines of the form "`? x y z`" to standard output. This asks for the value of the humidity at point (x, y, z). For each such line, the grader will in response write a single line with the integer $H[x, y, z]$, which can be read from standard input by your program.

After all these lines, your program must write out exactly one line of the form "`! x y z`" and terminate. This claims that the point (x, y, z) is a suitable location for Maximus according to the criterion above. The grader will provide no response to this output.

All values of x, y, z must obey $1 \leq x \leq N$, $1 \leq y \leq M$, $1 \leq z \leq K$. If they do not, or some line has an invalid format, or you ask for more than Q values, the grader will respond with `-1` and exit. If this happens your program should also exit. If it continues, it may incorrectly get a verdict of Runtime Error or Time Limit Exceeded.

You *must* make sure to flush standard output before reading the grader's response, or else your program will get judged as Time Limit Exceeded. This works as follows in the supported languages:

- Java: `System.out.println()` flushes automatically.
- Python: `print()` flushes automatically.
- C++: `cout << endl;` flushes, in addition to writing a newline. If using printf, `fflush(stdout)`.
- Pascal: `Flush(Output)`.

To help deal with this interaction, we provide optional helper code which you may copy into your program. A link to this code for all supported languages (C++, Pascal, Java, Python) can be found in the sidebar of the Kattis problem page. The helper code also uses fast input/output routines, which may be useful for Python and Java (only relevant for the last two test groups).

The grader will be *non-adaptive*; that is, each test case will have a fixed set of humidity values that do not depend on what measurements are performed by the program.

Constraints

Your solution will be tested on a set of test groups, each worth a number of points. Each test group contains a set of test cases. To get the points for a test group you need to solve all test cases in the test group. Your final score will be the maximum score of a single submission.

Group	Points	Limits
1	10	$M = K = 1$, $N = 1\,000\,000$, $Q = 10\,000$
2	22	$M = K = 1$, $N = 1\,000\,000$, $Q = 35$
3	12	$K = 1$, $N = M = 200$, $Q = 4\,000$
4	19	$K = 1$, $N = M = 1\,000$, $Q = 3\,500$
5	14	$N = M = K = 100$, $Q = 100\,000$
6	23	$N = M = K = 500$, $Q = 150\,000$

Sample dialogue

In Kattis there is one sample test case. In this sample, the box has dimensions $3 \times 1 \times 1$ and the humidity in the three cells is $\{10, 14, 13\}$. Below is an example dialogue, with the lines preceded by JUDGE being the output of Kattis (i.e. the input to your program), and the lines preceded by YOU being your program's output.

As 14 is indeed greater than or equal to the neighboring values (10 and 13), the location $(2, 1, 1)$ is a suitable location for Maximus, and you used three queries, which was the maximum allowed number ($Q = 3$). Thus, this dialogue gives Accepted on the sample.

```
JUDGE:   3 1 1 3
YOU:     ? 3 1 1
JUDGE:   13
YOU:     ? 2 1 1
JUDGE:   14
YOU:     ? 1 1 1
JUDGE:   10
YOU:     ! 2 1 1
```

Sample Input 1	Sample Output 1
3 1 1 3 123123 fixed 10 14 13	blah