A. The Contest

2 seconds, 256 megabytes

Pasha is participating in a contest on one well-known website. This time he wants to win the contest and will do anything to get to the first place!

This contest consists of n problems, and Pasha solves ith problem in a_i time units (his solutions are always correct). At any moment of time he can be thinking about a solution to only one of the problems (that is, he cannot be solving two problems at the same time). The time Pasha spends to send his solutions is negligible. **Pasha can send any number of solutions at the same moment.**

Unfortunately, there are too many participants, and the website is not always working. Pasha received the information that the website will be working only during m time periods, jth period is represented by its starting moment l_j and ending moment r_j. Of course, Pasha can send his solution only when the website is working. In other words, Pasha can send his solution at some moment T iff there exists a period x such that $l_x \le T \le r_x$.

Pasha wants to know his best possible result. We need to tell him the minimal moment of time by which he is able to have **solutions to all problems submitted**, if he acts optimally, or say that it's impossible no matter how Pasha solves the problems.

Input

The first line contains one integer n $(1 \le n \le 1000)$ — the number of problems. The second line contains n integers a_i $(1 \le a_i \le 10^5)$ — the time Pasha needs to solve ith problem.

The third line contains one integer m $(0 \le m \le 1000)$ — the number of periods of time when the website is working. Next m lines represent these periods. jth line contains two numbers l_j and r_j $(1 \le l_j < r_j \le 10^5)$ — the starting and the ending moment of jth period.

It is guaranteed that the periods are not intersecting and are given in chronological order, so for every $j > 1$ the condition $l_j > r_{j-1}$ is met.

Output

If Pasha can solve and submit all the problems before the end of the contest, print the minimal moment of time by which he can have all the solutions submitted.

Otherwise print "-1" (without brackets).

input

```
2
3 4
2
1 4
7 9
```

output

```
7
```

input

```
1
5
1
1 4
```

output

```
-1
```

input

```
1
5
1
1 5
```

output

```
5
```

In the first example Pasha can act like this: he solves the second problem in 4 units of time and sends it immediately. Then he spends 3 time units to solve the first problem and sends it 7 time units after the contest starts, because at this moment the website starts working again.

In the second example Pasha invents the solution only after the website stops working for the last time.

In the third example Pasha sends the solution exactly at the end of the first period.

B. The Golden Age

1 second, 256 megabytes

Unlucky year in Berland is such a year that its number n can be represented as $n = x^a + y^b$, where a and b are non-negative integer numbers.

For example, if $x = 2$ and $y = 3$ then the years `4` and `17` are *unlucky* ($4 = 2^0 + 3^1$, $17 = 2^3 + 3^2 = 2^4 + 3^0$) and year `18` isn't *unlucky* as there is no such representation for it.

Such interval of years that there are no *unlucky* years in it is called *The Golden Age*.

You should write a program which will find maximum length of *The Golden Age* which starts no earlier than the year l and ends no later than the year r. If all years in the interval $[l, r]$ are *unlucky* then the answer is `0`.

Input

The first line contains four integer numbers x, y, l and r ($2 \le x, y \le 10^{18}$, $1 \le l \le r \le 10^{18}$).

Output

Print the maximum length of *The Golden Age* within the interval $[l, r]$.

If all years in the interval $[l, r]$ are *unlucky* then print `0`.

input
2 3 1 10
output
1

input
3 5 10 22
output
8

input
2 3 3 5
output
0

In the first example the *unlucky* years are `2, 3, 4, 5, 7, 9` and `10`. So maximum length of *The Golden Age* is achived in the intervals $[1, 1]$, $[6, 6]$ and $[8, 8]$.

In the second example the longest *Golden Age* is the interval $[15, 22]$.

C. The Tag Game

1 second, 256 megabytes

Alice got tired of playing the tag game by the usual rules so she offered Bob a little modification to it. Now the game should be played on an undirected rooted tree of n vertices. Vertex `1` is the root of the tree.

Alice starts at vertex `1` and Bob starts at vertex x ($x \neq 1$). The moves are made in turns, Bob goes first. In one move one can either stay at the current vertex or travel to the neighbouring one.

The game ends when Alice goes to the same vertex where Bob is standing. Alice wants to minimize the total number of moves and Bob wants to maximize it.

You should write a program which will determine how many moves will the game last.

Input

The first line contains two integer numbers n and x ($2 \le n \le 2 \cdot 10^5$, $2 \le x \le n$).

Each of the next $n - 1$ lines contains two integer numbers a and b ($1 \le a, b \le n$) — edges of the tree. It is guaranteed that the edges form a valid tree.

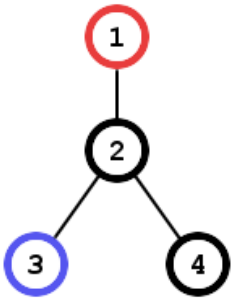
Output

Print the total number of moves Alice and Bob will make.

input
4 3 1 2 2 3 2 4
output
4

input
5 2 1 2 2 3 3 4 2 5
output
6

In the first example the tree looks like this:



The red vertex is Alice's starting position, the blue one is Bob's. Bob will make the game run the longest by standing at the vertex 3 during all the game. So here are the moves:

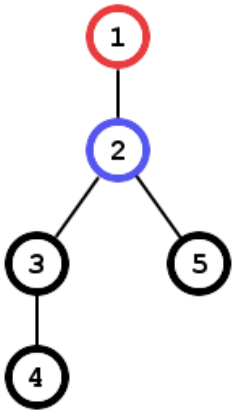
B: stay at vertex 3

A: go to vertex 2

B: stay at vertex 3

A: go to vertex 3

In the second example the tree looks like this:



The moves in the optimal strategy are:

B: go to vertex 3

A: go to vertex 2

B: go to vertex 4

A: go to vertex 3

B: stay at vertex 4

A: go to vertex 4

D. Two Melodies

2 seconds, 256 megabytes

Alice is a beginner composer and now she is ready to create another masterpiece. And not even the single one but two at the same time!

Alice has a sheet with n notes written on it. She wants to take two such non-empty non-intersecting subsequences that both of them form a *melody* and sum of their lengths is maximal.

Subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Subsequence forms a melody when each two adjacent notes either differs by 1 or are congruent modulo 7.

You should write a program which will calculate maximum sum of lengths of such two non-empty non-intersecting subsequences that both of them form a melody.

Input
The first line contains one integer number n ($2 \le n \le 5000$).

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^5$) — notes written on a sheet.

Output
Print maximum sum of lengths of such two non-empty non-intersecting subsequences that both of them form a melody.

input
4 1 2 4 5
output
4

input
6 62 22 60 61 48 49
output
5

In the first example subsequences $[1, 2]$ and $[4, 5]$ give length 4 in total.

In the second example subsequences $[62, 48, 49]$ and $[60, 61]$ give length 5 in total. If you choose subsequence $[62, 61]$ in the first place then the second melody will have maximum length 2, that gives the result of 4, which is not maximal.

E. Army Creation

2 seconds, 256 megabytes

As you might remember from our previous rounds, Vova really likes computer games. Now he is playing a strategy game known as Rage of Empires.

In the game Vova can hire n different warriors; ith warrior has the type a_i. Vova wants to create a *balanced* army hiring some subset of warriors. An army is called *balanced* if for each type of warrior present in the game there are not more than k warriors of this type in the army. Of course, Vova wants his army to be as large as possible.

To make things more complicated, Vova has to consider q different plans of creating his army. ith plan allows him to hire only warriors whose numbers are not less than l_i and not greater than r_i.

Help Vova to determine the largest size of a *balanced* army for each plan.

Be aware that the plans are given in a modified way. See input section for details.

Input
The first line contains two integers n and k ($1 \le n, k \le 100000$).

The second line contains n integers $a_1, a_2, ... a_n$ ($1 \le a_i \le 100000$).

The third line contains one integer q ($1 \le q \le 100000$).

Then q lines follow. ith line contains two numbers x_i and y_i which represent ith plan ($1 \le x_i, y_i \le n$).

You have to keep track of the answer to the last plan (let's call it $last$). In the beginning $last = 0$. Then to restore values of l_i and r_i for the ith plan, you have to do the following:

1. $l_i = ((x_i + last) \bmod n) + 1$;
2. $r_i = ((y_i + last) \bmod n) + 1$;
3. If $l_i > r_i$, swap l_i and r_i.

Output
Print q numbers. ith number must be equal to the maximum size of a *balanced* army when considering ith plan.

input

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

output

```
2
4
1
3
2
```

In the first example the real plans are:

1. 1 2
2. 1 6
3. 6 6
4. 2 4
5. 4 6

F. Bipartite Checking

6 seconds, 256 megabytes

You are given an undirected graph consisting of n vertices. Initially there are no edges in the graph. Also you are given q queries, each query either adds one undirected edge to the graph or removes it. After each query you have to check if the resulting graph is bipartite (that is, you can paint all vertices of the graph into two colors so that there is no edge connecting two vertices of the same color).

Input

The first line contains two integers n and q ($2 \le n, q \le 100000$).

Then q lines follow. ith line contains two numbers x_i and y_i ($1 \le x_i < y_i \le n$). These numbers describe ith query: if there is an edge between vertices x_i and y_i, then remove it, otherwise add it.

Output

Print q lines. ith line must contain `YES` if the graph is bipartite after ith query, and `NO` otherwise.

input

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

output

```
YES
YES
NO
YES
NO
```