Problem A
AGI

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

Artificial General Intelligence (AGI for short) seems increasingly inevitable. More and more people no longer ask *if* we will reach this level, but rather *when.* So far, n futurologists have presented their predictions regarding when AGI will appear. The i-th prediction is given as a time interval $[A_i, B_i)$, which means that according to it, AGI will emerge at a time t satisfying $A_i \le t < B_i$.

For each prediction you must decide whether you consider it to be true or not.

Your task is to make these decisions so that, regardless of the actual moment x at which AGI finally appears, you guarantee at least $\left\lfloor \frac{n-1}{2} \right\rfloor$ correct evaluations.

You may assume that the test cases are chosen in a way that guarantees at least one valid answer exists.

You have to solve t independent test cases.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. Each test case consists of a line with an integer n ($1 \le n \le 500\,000$) — the number of predictions, followed by n lines describing the predictions, each containing two integers A_i and B_i ($0 \le A_i < B_i \le 10^9$), denoting the start and end of the time interval of the i-th prediction.

The sum of all n across all test cases does not exceed $500\,000$.

Output

For each test case print a single line containing a string of length n consisting of the letters `T` and `N`. The j-th character of the string is your evaluation of the j-th prediction:

- `T` means you affirm this prediction.
- `N` means you reject this prediction.

If there are multiple answers satisfying the problem requirements, you may print any of them.

Example

For the input:

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

a correct output is for example:

```
NNNN
TNTNN
```

Explanation of the examples: In the first test case it suffices to correctly evaluate at least $\left\lfloor \frac{4-1}{2} \right\rfloor = 1$ prediction. The intersection of all intervals is empty, so by rejecting all predictions we will be correct on at least one evaluation.

In the second test case we must be correct on at least $\left\lfloor \frac{5-1}{2} \right\rfloor = 2$ evaluations. For example, if AGI appears at time $x = 2$, then predictions 1 and 2 would be true, so we correctly evaluated predictions $1, 4, 5$. If instead $x = 5.7$, then all predictions turn out to be true, so we correctly evaluated only predictions $1, 3$, which is still sufficient. One can prove that regardless of x, at least 2 predictions will be evaluated correctly.

Problem B
Collatz Sum

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 0.5 s. *21.06.2025*

We define the function f on natural numbers:

$$f(x) = \begin{cases} x/2, & \text{if } x \text{ is even,} \\ 3x+1, & \text{if } x \text{ is odd.} \end{cases}$$

Let $g(x)$ be the result of applying f to itself k times:

$$g(x) = \underbrace{f\big(f(\ldots f}_{k \text{ times}}(x)\ldots)\big).$$

Your task is, given N and k, to compute the sum

$$S = \sum_{x=1}^{N} g(x) \pmod{10^9+7}.$$

Input

A single line containing two integers N and k ($1 \le N \le 10^{12},\ 0 \le k \le 32$).

Output

Output a single number – the value of the sum S modulo $10^9 + 7$.

Example

For the input:

```
10 2
```

the correct output is:

```
73
```

Whereas for the input:

```
999888777666 1
```

the correct output is:

```
990122835
```

Explanation of the examples:

In the first example, the values of the function $g(i)$ for consecutive i are: $2, 4, 5, 1, 8, 10, 11, 2, 14, 16$. Their sum is 73.

In the second example, $S = 874805371740356549439861$.

Problem C
DNA Subsequences

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 3 s. *21.06.2025*

In this task, we consider sequences of nucleotides in a DNA molecule, which are strings composed of characters '`A`', '`C`', '`G`', and '`T`'. For each natural number k, there are 4^k different k-letter nucleotide sequences. For a fixed natural number k, we say that a given nucleotide sequence s is *k-rich* if all k-letter nucleotide sequences are subsequences of s (not necessarily contiguous).

You are given an n-letter nucleotide sequence s. For each natural number k in the range $[1, n]$, output the minimum number of characters that must be changed in s to make it a k-rich sequence. Note that for each k, the result is calculated independently.

Input

The first line of the input contains an integer n ($1 \leq n \leq 200\,000$), representing the length of the string s.

The second line of the input contains an n-letter nucleotide sequence s, consisting only of characters '`A`', '`C`', '`G`', and '`T`'.

Output

The output should consist of n integers; the k-th integer should represent the minimum number of characters that must be changed in s to make s a k-rich nucleotide sequence. If it is impossible to change the characters in s in the described way for a given k, then the k-th number should be -1 instead.

Example

For the input data:

```
8
AAGTAGAA
```

the correct result is:

```
1 3 -1 -1 -1 -1 -1 -1
```

Explanation: For $k = 1$, we can change s with one modification to, for example, `AAGTCGAA`. The resulting nucleotide sequence then contains all one-letter words as subsequences (in other words, each of the four letters appears at least once), and thus is 1-rich.

For $k = 2$, we can change s with three modifications to, for example, a 2-rich nucleotide sequence `CAGTTGAC`. Note that we could not change s to, for example, the sequence `CCGTTGAA`, as it does not contain the two-letter word `AC` as a subsequence.

For $k > 2$, it is impossible to make the sequence s k-rich.

Problem D
Fern Market

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 3 s. *21.06.2025*

A fern market will be open over the next n days in Bytown. The price of a fern on the i-th day is predetermined as a_i bytalars. Each day, you can decide to either buy or sell a fern at the given price, but (due to logistical issues) you can only buy or sell at most one fern per day. You can also choose to do nothing on any day. Naturally, you cannot sell ferns if you do not have any at that moment. However, you can store an unlimited number of ferns and wait for a good time to sell them. You have substantial savings and will not run out of funds for fern investments. You do not own any ferns at the start of day 1, and you also want to end up with no ferns left after the market closes on day n.

Define $f(a_1, a_2, \ldots, a_n)$ as the maximum possible profit (in bytalars) from trading ferns if you plan their purchase and sale optimally in advance. Then, for any natural number n, let $g(n)$ be the sum of $f(p)$ for all $n!$ permutations p of integers from 1 to n.

You are given two natural numbers k and m, where m is a prime number. Output the remainders from the division of each of the values $g(1), g(2), \ldots, g(k)$ modulo m.

Input

The first and only line of the input contains two integers k and m ($1 \leq k \leq 7\,000$; $10^8 + 7 \leq m \leq 10^9 + 7$; m is prime), as described in the problem statement.

Output

The output should consist of k lines. The i-th line should contain the remainder of the value $g(i)$ modulo m.

Example

For the input data:

```
4 1000000007
```

the correct result is:

```
0
1
8
64
```

Explanation: If $n = 1$, the market lasts only one day. Since you start without any ferns and want to end without any ferns, you cannot make any transactions.

If $n = 2$, there are two possible price sequences: $[1, 2]$ and $[2, 1]$. In the first case, you can buy a fern for one bytalar and then sell it for two bytalars, making a profit of one bytalar. Thus, $f(1, 2) = 1$. In the second case, you cannot make any profit, i.e., $f(2, 1) = 0$.

For $n = 3$, there are six possible price sequences:

- $f(1, 2, 3) = 2$,
- $f(1, 3, 2) = 2$,
- $f(2, 1, 3) = 2$,
- $f(2, 3, 1) = 1$,
- $f(3, 1, 2) = 1$,
- $f(3, 2, 1) = 0$.

In the first three cases, you can buy a fern for one bytalar and then sell it for three bytalars. In the fourth case, you can buy a fern for two bytalars and sell it for three. In the fifth case, you can buy a fern for one bytalar and sell it for two. In the last case, you cannot make any profit.

Note that you can store more than one fern. For example, $f(2, 1, 4, 3) = 4$ because you can buy ferns on the first and second days, and then sell them on the third and fourth days.

Problem E
One Bit

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

You are given a sequence $a_1, a_2, \ldots, a_n$ consisting of non-negative integers. A sequence is called *remarkable* if the binary representations of each two consecutive numbers differ by **at most** one bit[*]. You can choose any elements of the sequence and change them to any non-negative integers. What is the minimum number of elements that need to be changed to make the given sequence remarkable?

Input

The first line of the input contains an integer n ($1 \leq n \leq 300\,000$), representing the length of the sequence.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 2^{60} - 1$).

Output

The output should contain a single integer – the minimum number of elements in the sequence $a_1, a_2, \ldots, a_n$ that need to be changed to make this sequence remarkable.

Example

For the input data:

```
5
4 0 3 3 10
```

the correct result is:

```
2
```

Explanation: We can change, for example, the third and fourth elements, creating the remarkable sequence $[4, 0, 2, 10, 10]$. It can be shown that it is not possible to change only one number to make the sequence remarkable.

[*]When comparing the binary representations of two numbers of different lengths, we pad the shorter number with zeros at the front to match their lengths. For example, to compare $3_{(10)} = 11_{(2)}$ with $16_{(10)} = 10000_{(2)}$, we pad the first number with zeros at the front, creating the binary sequence 00011. The binary sequences 10000 and 00011 differ in three positions, thus the numbers 3 and 16 differ by three bits.

Problem F
Puzzle IV

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

You are given a sequence of n integers $p_1, p_2, \ldots, p_n$, initially a permutation of numbers from 1 to n. Your task is to sort it in ascending order. To achieve this, you can perform operations of two types: either add the value of an element to its adjacent element or subtract the value of an element from its adjacent element.

During the operations, the value of any element must not exceed the range $[1, n]$ at any point. The number of operations must not exceed 2 500 000.

Input

The first line of the input contains an integer n ($2 \leq n \leq 30\,000$), representing the length of the input sequence.

The second line of the input contains n pairwise distinct integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$) – the sequence to be sorted.

Output

The first line of the output should contain a single integer r ($0 \leq r \leq 2\,500\,000$), representing the number of operations you want to perform.

The next r lines should contain descriptions of the consecutive operations in the following format:

- `a + b`: add the value of the b-th element to the a-th element.
- `a - b`: subtract the value of the b-th element from the a-th element.

In the above operations, $|a - b| = 1$ must always hold.

Note that you do not have to minimize the number of operations. It can be proven that it is always possible to sort the permutation by performing at most 2 500 000 operations.

Example

For the input data:

```
3
1 3 2
```

the correct result is:

```
3
2 - 3
3 + 2
2 + 1
```

Explanation: The sequence becomes $[1, 1, 2]$ after the first operation, then $[1, 1, 3]$ after the second operation, and $[1, 2, 3]$ after the third operation.

Problem G
Furniture

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 0.5 s. *21.06.2025*

Bajtazar has ordered N new pieces of furniture for his apartment and wants to arrange them all in his living room, which is a rectangle of dimensions $A \times B$. Each piece of furniture is also a rectangle, the ith piece has dimensions $c_i \times d_i$.

Each piece must be placed against one of the walls of length A, with its side of length c_i flush against that wall. Of course, no two pieces of furniture may overlap; they may at most touch along sides or at corners.

Help Bajtazar by writing a program that determines whether it is possible to place all N pieces of furniture in his living room.

You have T independent test cases to solve.

Input

The first line contains a single integer T ($1 \leq T \leq 30$), the number of test cases.

In the first line of each test case, there are three integers A, B, N ($1 \leq A, B \leq 10^6$, $1 \leq N \leq 1000$) denoting the dimensions of Bajtazar's living room and the number of pieces of furniture he has ordered. Each of the next N lines contains two integers c_i, d_i ($1 \leq c_i \leq A$, $1 \leq d_i \leq B$), the dimensions of the i-th piece of furniture.

The sum of N over all test cases does not exceed 1000.

Output

Output T lines. On the i-th line output the word `TAK` (Polish for yes) if it is possible to place all the furniture in the i-th test case, or `NIE` (Polish for no) otherwise.

Example

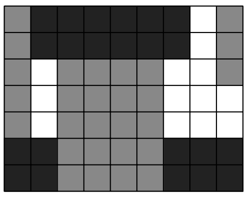
For the input data:

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

the correct result is:

```
TAK
NIE
TAK
NIE
```

Example Explanation: A valid arrangement of the furniture in the first test case is shown below:



Problem H
Merge Sort

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 3 s. *21.06.2025*

Merge sort is one of the most well-known sorting algorithms. In this task, we use an implementation described by the following pseudocode:

Algorithm: Implementation of the merge sort algorithm.

```
function MergeSort($p, n$)    ▷ Returns a sorted version of the list $p = [p[0], p[1], \ldots, p[n-1]]$.
    if $n = 1$ then
        return $p$
    left ← MergeSort($[p[0], \ldots, p[\lceil n/2 \rceil - 1]], \lceil n/2 \rceil$)
    right ← MergeSort($[p[\lceil n/2 \rceil], \ldots, p[n-1]], \lfloor n/2 \rfloor$)
    $(i, j) \leftarrow (0, 0)$
    result ← []    ▷ [] denotes an empty list.
    while $i <$ |left| and $j <$ |right| do    ▷ |left|, |right| are lengths of lists left, right.
        if left[$i$] < right[$j$] then
            Append the value left[$i$] to the end of list result.
            $i \leftarrow i + 1$
        else
            Append the value right[$j$] to the end of list result.
            $j \leftarrow j + 1$
    Append the values left[$i$], ..., left[|left| − 1] to the end of list result.
    Append the values right[$j$], ..., right[|right| − 1] to the end of list result.
    return result
```

You are given numbers n, a, and b. Your task is to count such permutations of numbers from 1 to n that sorting the permutation using the MergeSort procedure performs a comparison of a and b at least once in line 9. Output the remainder of the division of the number of such permutations by 1 000 000 007 ($10^9 + 7$).

Note that a permutation should be counted if numbers a and b are compared in any recursive call of the MergeSort procedure, not necessarily in the shallowest call. Also note that we allow comparison of numbers a and b in any order. In other words, checking only one condition $a < b$ or $b < a$ is enough to count the permutation.

Additionally, you need to solve multiple test cases.

Input

The first line of the input contains an integer t ($1 \le t \le 10\,000$), representing the number of test cases.

The next t lines contain descriptions of consecutive test cases. The i-th line contains three integers n, a, and b ($2 \le n \le 1\,000\,000; 1 \le a, b \le n; a \ne b$), as described in the task.

Output

The output should consist of t lines. The i-th line should contain a single integer – the remainder of the division by $10^9 + 7$ of the number of sought permutations in the i-th test case.

Example

For the input data:

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

the correct result is:

```
24
52
4
```

Explanation: In the first test case, the numbers 2 and 3 would be compared during the sorting of all permutations of length 4.

In the third test case, we will compare the numbers 1 and 3 during the sorting of permutations $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$ and $[3, 1, 2]$.

Problem I
Santa Claus

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 1 s. *21.06.2025*

Santa Claus has a tough job: not only does he have to deliver presents to all the children on time, but he also must make sure that each child gets the right gift. Good children receive toys, tickets to the Algorithmic Engagement finals, or a bicycle, while naughty ones get a lump of coal or a Windows Vista installation.

To stay on schedule, this evening Santa Claus must deliver presents to exactly K children, but the elves forgot to label the presents with the recipients' names! Santa must figure it out himself — choose which K out of N presents to deliver, and assign them to K out of M waiting children.

Santa knows the quality j_i of each present (which may be positive or nonpositive), and the niceness g_i of each child (which may also be positive or nonpositive).

Santa would like to maximize the fairness of the gift assignment: if present i is given to child k, then the total fairness increases by $j_i \cdot g_k$.

Can you help him determine the maximum possible fairness of assigning exactly K presents?

Input

The first line contains three integers K, N, and M ($1 \le K, N, M \le 200\,000$, $K \le \min(N, M)$), representing the number of presents Santa wants to deliver today, the number of presents in the warehouse, and the number of children waiting for a present, respectively.

The second line contains N integers $j_1, \ldots, j_N$ ($-10^6 \le j_i \le 10^6$), representing the quality of the presents.

The third line contains M integers $g_1, \ldots, g_M$ ($-10^6 \le g_i \le 10^6$), representing the niceness of the children.

Output

Output a single integer — the maximum possible fairness of assigning K presents.

Example

For the following input:

```
3 3 5
2 -2 4
-10 0 -2 3 2
```

the correct output is:

```
36
```

For the input:

```
1 4 4
0 0 0 0
2 4 -2 0
```

the correct output is:

```
0
```

And for the input:

```
1 2 2
1 2
-1 -2
```

the correct output is:

```
-1
```

Explanation of the examples:

In the first example, all presents should be distributed as follows:

- The present with quality 2 should be given to the child with niceness 2.
- The present with quality -2 should be given to the child with niceness -10.
- The present with quality 4 should be given to the child with niceness 3.

The total fairness is $2 \cdot 2 + (-2) \cdot (-10) + 4 \cdot 3 = 36$.

In the second example, all presents in the warehouse are completely mediocre, so it doesn't matter which one is chosen or to whom it is given — the fairness will always be 0.

In the third example, all the children are naughty, and all the presents are attractive. The best we can do is to give the least attractive present to the least naughty child.

Problem J
Housing Estate

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

The ByteBud company plans to build a housing estate filled with apartment blocks. ByteBud has already purchased a rectangular plot of land and divided it into n rows and m columns. This way, the company has split the land into $n \cdot m$ rectangular cells. For simplicity, let coordinates (i, j) denote the cell located in the i-th row and j-th column of the plot. They need to choose some cells (maybe none or all) and build an apartment block on each of them. Your task is to decide on which cells to build the apartment blocks and determine the height (in floors) of each block.

Creating the design is not straightforward! According to the official zoning regulation, the block on the cell with coordinates (i, j) can be at most $h_{i,j}$ floors high. In particular, if $h_{i,j} = 0$, it means that no block can be built on that cell. Additionally, ByteBud has already created a preliminary plan for the space between the blocks. Specifically, for each point on the plot where four cells meet, they have already planned the exact sum of the heights of the blocks located on these cells. You are given the numbers $s_{i,j}$ for each pair of indices $i \in \{1, \dots, n-1\}$, $j \in \{1, \dots, m-1\}$; such a number specifies the requirement that the blocks on the cells with coordinates $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$ must have a total of exactly $s_{i,j}$ floors.

There are many requirements, and ByteBud needs the housing estate design today... So, check if there exists a housing estate design that meets all the requirements – and if it exists, propose any valid one!

Input

The first line of the input contains two integers n and m ($2 \le n, m \le 300$), representing the number of rows and columns, respectively, into which ByteBud has divided the plot.

Then, the i-th of the following n lines contains m integers $h_{i,1}, \dots, h_{i,m}$ ($0 \le h_{i,j} \le 10^{10}$); the j-th number $h_{i,j}$ indicates the maximum height (in floors) of the apartment block on the cell with coordinates (i, j).

The next $n-1$ lines describe the preliminary space plan between the blocks. The i-th of those lines contains $m-1$ integers $s_{i,1}, \dots, s_{i,m-1}$ ($0 \le s_{i,j} \le 10^{10}$); the j-th number $s_{i,j}$ specifies the requirement for the exact sum of the heights (in floors) of the apartment blocks on the cells with coordinates $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$.

Output

If a housing estate design meeting all the requirements exists, print `TAK` in the first line of the output. Then provide an example of such a design in the following n lines. The i-th line should contain m non-negative integers $x_{i,1}, \dots, x_{i,m}$ indicating the heights (in floors) of the apartment blocks on the cells with coordinates $(i, 1), \dots, (i, m)$.

If the required design does not exist, print `NIE` in the only line of the output.

When printing `TAK` or `NIE`, every character can be lowercase or uppercase.

Example

For the input data:

```
3 4
5 7 9 4
3 4 9 6
9 0 8 5
11 26 20
13 16 18
```

a correct result is for example:

```
TAK
0 6 9 3
1 4 7 1
8 0 5 5
```

However, for the input data:

```
2 2
0 0
0 0
1000
```

the correct result is:

```
NIE
```

Problem K
Projects

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

You are a manager in a software company. Your n programmers have m projects scheduled for today. Project i can only be worked on by programmers a_i and b_i. Programmer a_i would finish the project in t_i byteseconds, and programmer b_i (if $a_i \neq b_i$) can work twice slower on this project. The two programmers can split the work: programmer a_i can choose a *real* value x_i ($0 \leq x_i \leq t_i$) and spend x_i byteseconds on project i. In that case, programmer b_i needs to spend $2 \cdot (t_i - x_i)$ byteseconds on this project. It may happen that $a_i = b_i$, in which case programmer a_i must complete the entire project on their own, spending t_i byteseconds.

Programmers a_i and b_i can work on project i independently. For example, they can work simultaneously or allocate their time to the project at completely different moments.

Each programmer can work on only one project at a time. The total time a programmer will work is the sum of the times spent on all projects. We assume that switching between projects takes negligible time.

All programmers will start work at the same time today. Some of them will tell you that they are in a hurry and want to leave work as early as possible. As a good manager, you want to meet these expectations. However, you do not yet know exactly which programmers are in a hurry. Therefore, you want to consider q independent scenarios. Each is described by a non-empty subset of programmers who are in a hurry. For each scenario, determine the smallest possible real number T such that you can schedule the work of the programmers so that all projects are eventually completed, and each programmer in a hurry works for no more than T byteseconds.

It can be proven that the results will be rational numbers. So provide all answers as irreducible fractions.

Input

The first line of the input contains three integers n, m and q ($1 \leq n \leq 13$; $1 \leq m \leq 200$; $1 \leq q \leq 10\,000$), representing the number of programmers, the number of projects, and the number of scenarios to consider, respectively.

Each of the next m lines contains three integers a_i, b_i and t_i ($1 \leq a_i, b_i \leq n$; $1 \leq t_i \leq 1\,000\,000$), representing the programmers responsible for a given project and the time required to complete the project by programmer a_i.

The next q lines describe the scenarios; the i-th scenario is one line that contains a binary string s_i of length n; the j-th character is '`1`' if the j-th programmer is in a hurry, and '`0`' otherwise. Each binary string s_i contains at least one character '`1`'.

Output

The output should consist of q lines; the i-th line should contain one rational number T written in irreducible fraction form x/y ($\mathrm{GCD}(x, y) = 1$ and $y > 0$) – the minimal possible limit on the work time for the programmers in a hurry in the i-th scenario.

Example

For the input data:

```
5 7 7
2 1 2
2 2 1
3 2 3
3 4 5
4 3 2
1 5 7
1 5 7
10000
01000
00110
11100
11111
01111
01111
```

the correct result is:

```
0/1
1/1
4/1
18/7
28/3
19/4
19/4
```

Explanation:

In the first scenario, programmer 1 can leave work immediately as the other programmers can handle the projects without him.

In the second scenario, programmer 2 must complete the second project, which will take him 1 bytesecond.

In the third scenario, programmer 4 will spend 2 byteseconds on project five and 2 byteseconds on project four. And programmer 3 should spend 4 byteseconds on project four.

In the fourth scenario, programmers 1 and 3 will each spend $\frac{18}{7}$ byteseconds on projects one and three, respectively. Programmer 2 needs to spend, respectively, $\frac{5}{7}$, 1 and $\frac{6}{7}$ byteseconds to finish the first three projects, finishing the work day at time $\frac{5+7+6}{7} = \frac{18}{7}$ too.

In the fifth scenario, programmers 1 and 5 each need to spend at least $\frac{28}{3}$ byteseconds on the last two projects. There is a strategy where each programmer finishes work after at most $\frac{28}{3}$ byteseconds.

Problem L
Directed Hanoi

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

The game board for the game of Directed Hanoi consists of N pegs numbered from 1 to N. Between some pairs of pegs there are directed edges, always leading from a peg with a lower number to a peg with a higher number. Initially, on peg 1 there are K disks sized from 1 to K, arranged (from bottom to top) from largest to smallest. The goal of the game is to move all the disks onto peg number N.

In a single move, you can take the disk that is on the top of some peg A and place it on the top of any other peg B, to which there exists a path from peg A (not necessarily a single edge), provided that peg B is either empty, or the disk currently on top of peg B is larger than the disk being moved. It does not matter what disks are on the pegs along the path from A to B.

Your task is, given a Directed Hanoi board, to determine the largest K for which there exists a way to achieve the game's goal (i.e., to move all K disks onto peg N). It can be proven that the maximum number of disks is always finite.

Input

The first line of input contains two integers: N $(2 \leq N \leq 500)$ and M $(0 \leq M \leq N(N-1)/2)$, representing the number of pegs and the number of edges, respectively.

Each of the next M lines describes one edge, consisting of two integers, a_i and b_i $(1 \leq a_i < b_i \leq N)$, meaning that pegs a_i and b_i are connected by a directed edge. Each pair of pegs appears at most once in the input.

Output

Output a single integer — the largest possible K such that when playing Directed Hanoi on the pegs described in the input, it is possible to move K disks from peg 1 to peg N.

Example

For the input:

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

the correct output is:

```
5
```

For the input:

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

the correct output is:

```
5
```

For the input:

```
2 0
```

the correct output is:

```
0
```

Explanation of examples:

In the first example, a sample sequence of moves transferring 5 disks from peg 1 to peg 5 looks like this:

Peg 1	54321	5432	543	543	54	5						
Peg 2		1	1		3	3	3	3	3			
Peg 3			2	21	21	21	21	21	2	2		
Peg 4						4	4		1	1	1	
Peg 5							5	54	54	543	5432	54321

In the third example, there are no edges, so it is not possible to move any disk.

Problem M
Clubs

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 1 s. *21.06.2025*

Clubs is a game for two players who cooperate to achieve an *optimal contract*. Each of them holds cards, some of which are clubs. Each player knows how many clubs they have, but not how many the other player has. The optimal contract depends on the total number of clubs both players have in hand — if it is **less** than X, then the optimal contract is N, otherwise it is $N + 1$.

The contract is reached through bidding. The first player starts by proposing a contract — a positive integer. Then the players alternate turns. On their turn, a player may either accept the last proposed contract or propose a different, higher contract (also a positive integer). The bidding ends when one of the players accepts a contract; it is a success if this is the optimal contract.

Algosia and Bajtek are playing Clubs. They both know X, N, and that the number of clubs in Algosia's hand is one of the values $a_1, a_2, \ldots, a_k$, while the number of clubs in Bajtek's hand is one of the values $b_1, b_2, \ldots, b_l$. Just before the game begins, each of them learns the exact number of clubs in their own hand.

They are interested in finding the minimum N for which there exists a bidding strategy that guarantees success. Algosia is always the first player. It can be proven that such a minimal value of N exists.

Input

The first line contains three integers X, k, and l ($1 \le X \le 10^9$; $1 \le k, l \le 1000$), representing respectively the total number of clubs in both players' hands required for the optimal contract to be $N + 1$, and the number of possible club counts in Algosia's and Bajtek's hands.

The second line contains k integers $a_1, \ldots, a_k$ ($1 \le a_1 < a_2 < \ldots < a_k \le 10^9$), representing the possible numbers of clubs in Algosia's hand. The third line contains l integers $b_1, \ldots, b_l$ ($1 \le b_1 < b_2 < \ldots < b_l \le 10^9$), representing the possible numbers of clubs in Bajtek's hand.

Output

Print a single integer — the smallest N such that there exists a strategy enabling Algosia and Bajtek to guarantee reaching the optimal contract.

Example

For the following input:

```
13 5 3
1 5 7 10 12
2 6 9
```

the correct output is:

```
3
```

Whereas for the input:

```
2 1 1
1
1
```

the correct output is:

```
0
```

Explanation of the examples:

In the first example, for $N = 3$, the following strategy may be used:

- Algosia proposes the contract 1 if she has 5, 7, or 10 clubs.
  - Bajtek responds with 2 if he has 6 clubs.
    - Algosia responds with 3 if she has 5 clubs. Bajtek accepts (They have 11 clubs in total).
    - Algosia responds with 4 if she has 7 or 10 clubs. Bajtek accepts (They have 13 or 16 clubs in total).
  - Bajtek responds with 3 if he has 2 clubs. Algosia accepts (They have 7, 9, or 12 clubs in total).
  - Bajtek responds with 4 if he has 9 clubs. Algosia accepts (They have 14, 16, or 19 clubs in total).
- Algosia proposes 2 if she has 1 or 12 clubs. Regardless of his club count, Bajtek responds with 3.
  - Algosia accepts if she has 1 club (They have 3, 7, or 10 clubs in total).
  - Algosia responds with 4 if she has 12 clubs. Bajtek accepts (They have 14, 18, or 21 clubs in total).

It can be proven that no suitable strategy exists for $N = 2$.

In the second example, for $N = 0$, Algosia bids $1 = N + 1$, and Bajtek accepts the contract.

Problem N
Frequency Function

The 3rd Universal Cup, Stage 40: Potyczki. Limits: 1024 MB, 2 s. *21.06.2025*

Let's define a function $f(a)$ which takes as an argument a sequence of n integers $a_1, a_2, \dots a_n$ in the range $[0, n]$ and returns a sequence $b_1, b_2, \dots b_n$ such that b_i is the number of occurrences of the number i in the sequence $a_1, a_2, \dots a_n$.

Additionally, let's define its k-fold composition:

$$f^k(a) = \begin{cases} a & \text{for } k = 0, \\ f(f^{k-1}(a)) & \text{for } k > 0. \end{cases}$$

You are given a sequence $a_1, a_2, \dots a_n$. Your task is to handle two types of queries:

- $1\ v\ x$ – Change the value of a_v to x.
- $2\ k\ v$ – Print the v-th element of the sequence $f^k(a)$.

Input

The first line of the input contains two integers n and q ($1 \leq n \leq 300\,000; 1 \leq q \leq 500\,000$), representing the length of the input sequence and the number of queries, respectively.

The second line of the input contains a sequence of n integers $a_1, a_2, \dots, a_n$ ($0 \leq a_i \leq n$).

The next q lines contain descriptions of the queries in the format specified in the problem statement. It holds that $1 \leq v \leq n$, $0 \leq x \leq n$ and $0 \leq k \leq 300\,000$.

It is guaranteed that there will be at least one query of the second type.

Output

The output should contain as many lines as there are queries of the second type. The i-th line should contain a single integer – the answer to the i-th query of the second type.

Example

For the input data:

```
6 6
2 1 2 3 0 3
2 3 2
1 5 2
2 0 2
1 2 3
2 0 2
2 2 3
```

the correct result is:

```
1
1
3
2
```

Explanation: Let's analyze the last query. We have:

- $f^0(a) = [2, 3, 2, 3, 2, 3]$,
- $f^1(a) = [0, 3, 3, 0, 0, 0]$,
- $f^2(a) = [0, 0, 2, 0, 0, 0]$.

The answer to this query is 2.