Codeforces Round 950 (Div. 3)

The problem statement has recently been changed. View the changes. ×

A. Problem Generator

1 second, 256 megabytes

Vlad is planning to hold m rounds next month. Each round should contain one problem of difficulty levels '`A`', '`B`', '`C`', '`D`', '`E`', '`F`', and '`G`'.

Vlad already has a bank of n problems, where the i-th problem has a difficulty level of a_i. There may not be enough of these problems, so he may have to come up with a few more problems.

Vlad wants to come up with as few problems as possible, so he asks you to find the minimum number of problems he needs to come up with in order to hold m rounds.

For example, if $m = 1$, $n = 10$, $a =$ '`BGECDCBDED`', then he needs to come up with two problems: one of difficulty level '`A`' and one of difficulty level '`F`'.

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n \le 50$, $1 \le m \le 5$) — the number of problems in the bank and the number of upcoming rounds, respectively.

The second line of each test case contains a string a of n characters from '`A`' to '`G`' — the difficulties of the problems in the bank.

Output

For each test case, output a single integer — the minimum number of problems that need to come up with to hold m rounds.

input
```
3
10 1
BGECDCBDED
10 2
BGECDCBDED
9 1
BBCDEFFGG
```

output
```
2
5
1
```

B. Choosing Cubes

1 second, 256 megabytes

Dmitry has n cubes, numbered from left to right from 1 to n. The cube with index f is his favorite.

Dmitry threw all the cubes on the table, and the i-th cube showed the value a_i $(1 \le a_i \le 100)$. After that, he arranged the cubes in non-increasing order of their values, from largest to smallest. If two cubes show the same value, they can go in any order.

After sorting, Dmitry removed the first k cubes. Then he became interested in whether he removed his favorite cube (note that its position could have changed after sorting).

For example, if $n = 5$, $f = 2$, $a = [4, 3, 3, 2, 3]$ (the favorite cube is highlighted in green), and $k = 2$, the following could have happened:

- After sorting $a = [4, 3, 3, 3, 2]$, since the favorite cube ended up in the second position, it will be removed.
- After sorting $a = [4, 3, 3, 3, 2]$, since the favorite cube ended up in the third position, it will not be removed.

Input

The first line contains an integer t $(1 \le t \le 1000)$ — the number of test cases. Then follow the descriptions of the test cases.

The first line of each test case description contains three integers n, f, and k $(1 \le f, k \le n \le 100)$ — the number of cubes, the index of Dmitry's favorite cube, and the number of removed cubes, respectively.

The second line of each test case description contains n integers a_i ($1 \le a_i \le 100$) — the values shown on the cubes.

Output

For each test case, output one line — "`YES`" if the cube will be removed in all cases, "`NO`" if it will not be removed in any case, "`MAYBE`" if it may be either removed or left.

You can output the answer in any case. For example, the strings "`YES`", "`nO`", "`mAyBe`" will be accepted as answers.

input
```
12
5 2 2
4 3 3 2 3
5 5 3
4 2 1 3 5
5 5 2
5 2 4 1 3
5 5 5
1 2 5 4 3
5 5 4
3 1 2 4 5
5 5 5
4 3 2 1 5
6 5 3
1 2 3 1 2 3
10 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1
42
5 2 3
2 2 1 1 2
2 1 1
2 1
5 3 1
3 3 2 3 2
```

output

```
MAYBE
YES
NO
YES
YES
YES
MAYBE
MAYBE
YES
YES
YES
NO
```

C. Sofia and the Lost Operations

2 seconds, 256 megabytes

Sofia had an array of n integers $a_1, a_2, \ldots, a_n$. One day she got bored with it, so she decided to **sequentially** apply m modification operations to it.

Each modification operation is described by a pair of numbers $\langle c_j, d_j\rangle$ and means that the element of the array with index c_j should be assigned the value d_j, i.e., perform the assignment $a_{c_j} = d_j$. After applying **all** modification operations **sequentially**, Sofia discarded the resulting array.

Recently, you found an array of n integers $b_1, b_2, \ldots, b_n$. You are interested in whether this array is Sofia's array. You know the values of the original array, as well as the values $d_1, d_2, \ldots, d_m$. The values $c_1, c_2, \ldots, c_m$ turned out to be lost.

Is there a sequence $c_1, c_2, \ldots, c_m$ such that the **sequential** application of modification operations $\langle c_1, d_1, \rangle, \langle c_2, d_2, \rangle, \ldots, \langle c_m, d_m\rangle$ to the array $a_1, a_2, \ldots, a_n$ transforms it into the array $b_1, b_2, \ldots, b_n$?

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

Then follow the descriptions of the test cases.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the original array.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the elements of the found array.

The fourth line contains an integer m ($1 \le m \le 2 \cdot 10^5$) — the number of modification operations.

The fifth line contains m integers $d_1, d_2, \ldots, d_m$ ($1 \le d_j \le 10^9$) — the preserved value for each modification operation.

It is guaranteed that the sum of the values of n for all test cases does not exceed $2 \cdot 10^5$, similarly the sum of the values of m for all test cases does not exceed $2 \cdot 10^5$.

Output

Output t lines, each of which is the answer to the corresponding test case. As an answer, output "`YES`" if there exists a suitable sequence $c_1, c_2, \ldots, c_m$, and "`NO`" otherwise.

You can output the answer in any case (for example, the strings "`yEs`", "`yes`", "`Yes`" and "`YES`" will be recognized as a positive answer).

input

```
7
3
1 2 1
1 3 2
4
1 3 1 2
4
1 2 3 5
2 1 3 5
2
2 3
5
7 6 1 10 10
3 6 1 11 11
3
4 3 11
4
3 1 7 8
2 2 7 10
5
10 3 2 2 1
5
5 7 1 7 9
4 10 1 2 9
8
1 1 9 8 7 2 10 4
4
1000000000 203 203 203
203 1000000000 203 1000000000
2
203 1000000000
1
1
1
5
1 3 4 5 1
```

output

```
YES
NO
NO
NO
YES
NO
YES
```

D. GCD-sequence

2 seconds, 256 megabytes

GCD (Greatest Common Divisor) of two integers x and y is the maximum integer z by which both x and y are divisible. For example, $GCD(36, 48) = 12$, $GCD(5, 10) = 5$, and $GCD(7, 11) = 1$.

Kristina has an array a consisting of exactly n positive integers. She wants to count the GCD of each neighbouring pair of numbers to get a new array b, called *GCD-sequence*.

So, the elements of the GCD-sequence b will be calculated using the formula $b_i = GCD(a_i, a_{i+1})$ for $1 \le i \le n-1$.

Determine whether it is possible to remove **exactly one** number from the array a so that the GCD sequence b is non-decreasing (i.e., $b_i \le b_{i+1}$ is always true).

For example, let Khristina had an array a = $[20, 6, 12, 3, 48, 36]$. If she removes $a_4 = 3$ from it and counts the GCD-sequence of b, she gets:

- $b_1 = GCD(20, 6) = 2$
- $b_2 = GCD(6, 12) = 6$
- $b_3 = GCD(12, 48) = 12$
- $b_4 = GCD(48, 36) = 12$

The resulting GCD sequence b = $[2, 6, 12, 12]$ is non-decreasing because $b_1 \leq b_2 \leq b_3 \leq b_4$.

Input

The first line of input data contains a single number t ($1 \leq t \leq 10^4$) — he number of test cases in the test.

This is followed by the descriptions of the test cases.

The first line of each test case contains a single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the number of elements in the array a.

The second line of each test case contains exactly n integers a_i ($1 \leq a_i \leq 10^9$) — the elements of array a.

It is guaranteed that the sum of n over all test case does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single line:

- "YES" if you can remove **exactly one** number from the array a so that the GCD-sequence of b is non-decreasing;
- "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes", and "YES" will all be recognized as a positive answer).

input

```
12
6
20 6 12 3 48 36
4
12 6 3 4
3
10 12 3
5
32 16 8 4 2
5
100 50 2 10 20
4
2 4 8 1
10
7 4 6 2 4 5 1 4 2 8
7
5 9 6 8 5 9 2
6
11 14 8 12 9 3
9
5 7 3 10 6 3 12 6 3
3
4 2 4
8
1 6 11 12 6 12 3 6
```

output

```
YES
NO
YES
NO
YES
YES
NO
YES
YES
YES
YES
YES
```

The first test case is explained in the problem statement.

E. Permutation of Rows and Columns

3 seconds, 256 megabytes

You have been given a matrix a of size n by m, containing a permutation of integers from 1 to $n \cdot m$.

A permutation of n integers is an array containing all numbers from 1 to n exactly once. For example, the arrays $[1]$, $[2, 1, 3]$, $[5, 4, 3, 2, 1]$ are permutations, while the arrays $[1, 1]$, $[100]$, $[1, 2, 4, 5]$ are not.

A matrix contains a permutation if, when all its elements are written out, the resulting array is a permutation. Matrices $[[1, 2], [3, 4]]$, $[[1]]$, $[[1, 5, 3], [2, 6, 4]]$ contain permutations, while matrices $[[2]]$, $[[1, 1], [2, 2]]$, $[[1, 2], [100, 200]]$ do not.

You can perform one of the following two actions in one operation:

- choose columns c and d ($1 \leq c, d \leq m$, $c \neq d$) and swap these columns;
- choose rows c and d ($1 \leq c, d \leq n$, $c \neq d$) and swap these rows.

You can perform any number of operations.

You are given the original matrix a and the matrix b. Your task is to determine whether it is possible to transform matrix a into matrix b using the given operations.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$) — the number of test cases. The descriptions of the test cases follow.

The first line of each test case description contains 2 integers n and m ($1 \leq n, m \leq n \cdot m \leq 2 \cdot 10^5$) — the sizes of the matrix.

The next n lines contain m integers a_{ij} each ($1 \leq a_{ij} \leq n \cdot m$). It is guaranteed that matrix a is a permutation.

The next n lines contain m integers b_{ij} each ($1 \leq b_{ij} \leq n \cdot m$). It is guaranteed that matrix b is a permutation.

It is guaranteed that the sum of the values $n \cdot m$ for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output "YES" if the second matrix can be obtained from the first, and "NO" otherwise.

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

input

```
7
1 1
1
1
2 2
1 2
3 4
4 3
2 1
2 2
1 2
3 4
4 3
1 2
3 4
1 5 9 6
12 10 4 8
7 11 3 2
1 5 9 6
12 10 4 8
7 11 3 2
3 3
1 5 9
6 4 2
3 8 7
9 5 1
2 4 6
7 8 3
2 3
1 2 6
5 4 3
6 1 2
3 4 5
1 5
5 1 2 3 4
4 2 5 1 3
```

output

```
YES
YES
NO
YES
YES
NO
YES
```

In the second example, the original matrix looks like this:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

By swapping rows 1 and 2, it becomes:

$$\begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix}$$

By swapping columns 1 and 2, it becomes equal to matrix b:

$$\begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$$

F1. Field Division (easy version)

3 seconds, 256 megabytes

This is an easy version of the problem; it differs from the hard version only by the question. The easy version only needs you to print whether some values are non-zero or not. The hard version needs you to print the exact values.

Alice and Bob are dividing the field. The field is a rectangle of size $n \times m$ ($2 \le n, m \le 10^9$), the rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to m from left to right. The cell at the intersection of row r and column c is denoted as (r, c).

Bob has k ($2 \le k \le 2 \cdot 10^5$) fountains, all of them are located in different cells of the field. Alice is responsible for dividing the field, but she must meet several conditions:

- To divide the field, Alice will start her path in any free (without a fountain) cell on the left or top side of the field and will move, each time moving to the adjacent cell **down** or **right**. Her path will end on the right or bottom side of the field.
- Alice's path will divide the field into two parts — one part will belong to Alice (this part includes the cells of her path), the other part — to Bob.
- Alice will own the part that includes the cell $(n, 1)$.
- Bob will own the part that includes the cell $(1, m)$.

Alice wants to divide the field in such a way as to get as many cells as possible.

Bob wants to keep ownership of all the fountains, but he can give one of them to Alice. First, output the integer α — the maximum possible size of Alice's plot, if Bob does not give her any fountain (i.e., all fountains will remain on Bob's plot). Then output k non-negative integers $a_1, a_2, \dots, a_k$, where:

- $a_i = 0$, if after Bob gives Alice the i-th fountain, the maximum possible size of Alice's plot does not increase (i.e., remains equal to α);
- $a_i = 1$, if after Bob gives Alice the i-th fountain, the maximum possible size of Alice's plot increases (i.e., becomes greater than α).

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains three integers n, m, and k ($2 \le n, m \le 10^9$, $2 \le k \le 2 \cdot 10^5$) — the field sizes and the number of fountains, respectively.

Then follow k lines, each containing two numbers r_i and c_i ($1 \le r_i \le n$, $1 \le c_i \le m$) — the coordinates of the cell with the i-th fountain. It is guaranteed that all cells are distinct and none of them is $(n, 1)$.

It is guaranteed that the sum of k over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, first output α — the maximum size of the plot that can belong to Alice if Bob does not give her any of the fountains. Then output k non-negative integers $a_1, a_2, \dots, a_k$, where:

- $a_i = 0$, if after Bob gives Alice the i-th fountain, the maximum possible size of Alice's plot does not increase compared to the case when all k fountains belong to Bob;
- $a_i = 1$, if after Bob gives Alice the i-th fountain, the maximum possible size of Alice's plot increases compared to the case when all k fountains belong to Bob.

If you output any other positive number instead of 1 that fits into a 64-bit signed integer type, it will also be recognized as 1. Thus, a solution to the hard version of this problem will also pass the tests for the easy version.

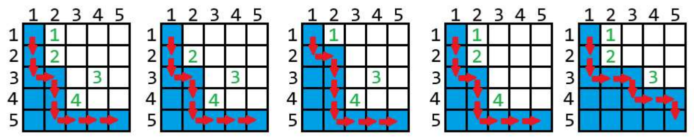
input

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

output

```
1
1 0 1
11
0 1 0 1
1
0 0 1 1 0 0 0 0 0
6
1 0 0 0
1
1 1 0 0 0
```

Below are the images for the second example:



The indices of the fountains are labeled in green. The cells belonging to Alice are marked in blue.

Note that if Bob gives Alice fountain 1 or fountain 3, then that fountain cannot be on Alice's plot.

F2. Field Division (hard version)

3 seconds, 256 megabytes

This is a hard version of the problem; it differs from the easy version only by the question. The easy version only needs you to print whether some values are non-zero or not. The hard version needs you to print the exact values.

Alice and Bob are dividing the field. The field is a rectangle of size $n \times m$ ($2 \le n, m \le 10^9$); the rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to m from left to right. The cell at the intersection of row r and column c is denoted as (r, c).

Bob has k ($2 \le k \le 2 \cdot 10^5$) fountains, all of them are located in different cells of the field. Alice is responsible for dividing the field, but she must meet several conditions:

- To divide the field, Alice will start her path in any free (without a fountain) cell on the left or top side of the field and will move, each time moving to the adjacent cell **down** or **right**. Her path will end on the right or bottom side of the field.
- Alice's path will divide the field into two parts — one part will belong to Alice (this part includes the cells of her path), the other part — to Bob.
- Alice will own the part that includes the cell $(n, 1)$.
- Bob will own the part that includes the cell $(1, m)$.

Alice wants to divide the field in such a way as to get as many cells as possible.

Bob wants to keep ownership of all the fountains, but he can give one of them to Alice. First, output the integer α — the maximum possible size of Alice's plot, if Bob does not give her any fountain (i.e., all fountains will remain on Bob's plot).

Then output k non-negative integers $a_1, a_2, \dots, a_k$, where a_i is a value such that after Bob gives Alice the i-th fountain, the maximum size of her plot will be $\alpha + a_i$.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains three integers n, m, and k ($2 \le n, m \le 10^9$, $2 \le k \le 2 \cdot 10^5$) — the field sizes and the number of fountains, respectively.

Then follow k lines, each containing two numbers r_i and c_i ($1 \le r_i \le n$, $1 \le c_i \le m$) — the coordinates of the cell with the i-th fountain. It is guaranteed that all cells are distinct and none of them is $(n, 1)$.

It is guaranteed that the sum of k over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, first output α — the maximum size of the plot that can belong to Alice if Bob does not give her any of the fountains. Then output k non-negative integers $a_1, a_2, \dots, a_k$, where a_i is a value such that after Bob gives Alice the i-th fountain, the maximum size of her plot will be $\alpha + a_i$.

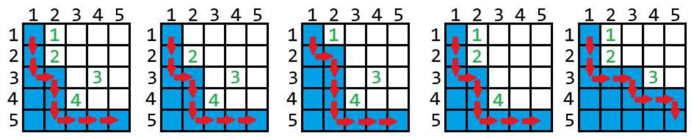
input

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

output

```
1
1 0 1
11
0 1 0 4
1
0 0 1 1 0 0 0 0 0
6
15 0 0 0
1
2 3 0 0 0
```

Below are the images for the second example:



The indices of the fountains are labeled in green. The cells belonging to Alice are marked in blue.

Note that if Bob gives Alice fountain 1 or fountain 3, then that fountain cannot be on Alice's plot.

G. Yasya and the Mysterious Tree

2.5 seconds, 512 megabytes

Yasya was walking in the forest and accidentally found a tree with n vertices. A tree is a connected undirected graph with no cycles.

Next to the tree, the girl found an ancient manuscript with m queries written on it. The queries can be of two types.

The first type of query is described by the integer y. The weight of **each** edge in the tree is replaced by the bitwise exclusive OR of the weight of that edge and the integer y.

The second type is described by the vertex v and the integer x. Yasya chooses a vertex u ($1 \le u \le n$, $u \neq v$) and mentally draws a bidirectional edge of weight x from v to u in the tree.

Then Yasya finds a simple cycle in the resulting graph and calculates the bitwise exclusive OR of all the edges in it. She wants to choose a vertex u such that the calculated value is **maximum**. This calculated value will be the answer to the query. It can be shown that such a cycle exists and is unique under the given constraints (independent of the choice of u). If an edge between v and u already existed, a simple cycle is the path $v \to u \to v$.

Note that the second type of query is performed *mentally*, meaning the tree does **not** change in any way after it.

Help Yasya answer all the queries.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The descriptions of the test cases follow.

The first line of each test case contains two integers n, m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$) — the number of vertices in the tree and the number of queries.

The next $n-1$ lines of each test case contain three integers v, u, w ($1 \le v, u \le n$, $1 \le w \le 10^9$) — the ends of some edge in the tree and its weight.

It is guaranteed that the given set of edges forms a tree.

The next m lines of each test case describe the queries:

- ^ y ($1 \le y \le 10^9$) — parameter of the first type query;
- ? v x ($1 \le v \le n$, $1 \le x \le 10^9$) — parameters of the second type query.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$. The same is guaranteed for m.

Output

For each test case, output the answers to the queries of the second type.

input

```
2
3 7
1 2 1
3 1 8
^ 5
? 2 9
^ 1
? 1 10
^ 6
? 3 1
? 2 9
5 6
1 2 777
3 2 2812
4 1 16
5 3 1000000000
^ 4
? 3 123
? 5 1000000000
^ 1000000000
? 1 908070
? 2 1
```

output

```
13 15 11 10
1000000127 2812 999756331 999999756
```

input

```
3
8 4
8 6 3
6 3 4
2 5 4
7 6 2
7 1 10
4 1 4
5 1 2
^ 4
^ 7
? 7 8
? 4 10
5 6
3 1 4
2 3 9
4 3 6
5 2 10
? 5 7
^ 1
^ 8
? 4 10
? 1 9
? 3 6
4 2
2 1 4
4 3 5
2 3 4
^ 13
? 1 10
```

output

```
14 13
13 8 11 11
10
```