Codeforces Round #732 (Div. 1)

A. AquaMoon and Strange Sort

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

AquaMoon has n friends. They stand in a row from left to right, and the i-th friend from the left wears a T-shirt with a number a_i written on it. Each friend has a direction (left or right). In the beginning, the direction of each friend is **right**.

AquaMoon can make some operations on friends. On each operation, AquaMoon can choose two **adjacent** friends and swap their positions. After each operation, the direction of both chosen friends will also be flipped: left to right and vice versa.

AquaMoon hopes that after some operations, the numbers written on the T-shirt of n friends in the row, read from left to right, become **non-decreasing**. Also she wants, that all friends will have a direction of **right** at the end. Please find if it is possible.

Input

The input consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 50)$ — the number of test cases.

The first line of each test case contains a single integer n $(1 \le n \le 10^5)$ — the number of Aquamoon's friends.

The second line contains n integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^5)$ — the numbers, written on the T-shirts.

It is guaranteed that the sum of n for all test cases does not exceed 10^5.

Output

For each test case, if there exists a possible sequence of operations, print "YES" (without quotes); otherwise, print "NO" (without quotes).

You can print each letter in any case (upper or lower).

Example

input

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

output

```
YES
YES
NO
```

Note

The possible list of operations in the first test case:

1. Swap a_1 and a_2. The resulting sequence is $3, 4, 2, 5$. The directions are: left, left, right, right.
2. Swap a_2 and a_3. The resulting sequence is $3, 2, 4, 5$. The directions are: left, left, right, right.
3. Swap a_1 and a_2. The resulting sequence is $2, 3, 4, 5$. The directions are: right, right, right, right.

B. AquaMoon and Chess

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Cirno gave AquaMoon a chessboard of size $1 \times n$. Its cells are numbered with integers from 1 to n from left to right. In the beginning, some of the cells are occupied with at most one pawn, and other cells are unoccupied.

In each operation, AquaMoon can choose a cell i with a pawn, and do **either** of the following (if possible):

- Move pawn from it to the $(i+2)$-th cell, if $i+2 \le n$ and the $(i+1)$-th cell is occupied and the $(i+2)$-th cell is unoccupied.
- Move pawn from it to the $(i-2)$-th cell, if $i-2 \ge 1$ and the $(i-1)$-th cell is occupied and the $(i-2)$-th cell is unoccupied.

You are given an initial state of the chessboard. AquaMoon wants to count the number of states reachable from the initial state with some sequence of operations. But she is not good at programming. Can you help her? As the answer can be large find it modulo

$998\,244\,353$.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases.

The first line contains a single integer n ($1 \le n \le 10^5$) — the size of the chessboard.

The second line contains a string of n characters, consists of characters "0" and "1". If the i-th character is "1", the i-th cell is initially occupied; otherwise, the i-th cell is initially unoccupied.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print the number of states that reachable from the initial state with some sequence of operations modulo $998\,244\,353$.

Example

input

```
6
4
0110
6
011011
5
01010
20
10001111110110111000
20
00110110100110111101
20
11101111011000100010
```

output

```
3
6
1
1287
1287
715
```

Note

In the first test case the strings "1100", "0110" and "0011" are reachable from the initial state with some sequence of operations.

C. AquaMoon and Permutations

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Cirno has prepared n arrays of length n each. Each array is a permutation of n integers from 1 to n. These arrays are special: for all $1 \le i \le n$, if we take the i-th element of each array and form another array of length n with these elements, the resultant array is also a permutation of n integers from 1 to n. In the other words, if you put these n arrays under each other to form a matrix with n rows and n columns, this matrix is a Latin square.

Afterwards, Cirno added additional n arrays, each array is a permutation of n integers from 1 to n. For all $1 \le i \le n$, there exists **at least one** position $1 \le k \le n$, such that for the i-th array and the $(n+i)$-th array, the k-th element of both arrays is the same. Notice that the arrays indexed from $n+1$ to $2n$ **don't have to** form a Latin square.

Also, Cirno made sure that for all $2n$ arrays, no two arrays are completely equal, i. e. for all pair of indices $1 \le i < j \le 2n$, there exists **at least one** position $1 \le k \le n$, such that the k-th elements of the i-th and j-th array are **different**.

Finally, Cirno arbitrarily changed the order of $2n$ arrays.

AquaMoon calls a subset of all $2n$ arrays of size n **good** if these arrays from a Latin square.

AquaMoon wants to know how many good subsets exist. Because this number may be particularly large, find it modulo $998\,244\,353$. Also, she wants to find any good subset. Can you help her?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a single integer n ($5 \le n \le 500$).

Then $2n$ lines followed. The i-th of these lines contains n integers, representing the i-th array.

It is guaranteed, that the sum of n over all test cases does not exceed 500.

Output

For each test case print two lines.

In the first line, print the number of good subsets by modulo $998\,244\,353$.

In the second line, print n indices from 1 to $2n$ — indices of the n arrays that form a good subset (you can print them in any order). If there are several possible answers — print any of them.

Example

input

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

output

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

Note

In the first test case, the number of good subsets is 1. The only such subset is the set of arrays with indices 1, 2, 3, 4, 5, 6, 7.

In the second test case, the number of good subsets is 2. They are 1, 3, 5, 6, 10 or 2, 4, 7, 8, 9.

D. AquaMoon and Wrong Coordinate

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Cirno gives AquaMoon a problem. There are m people numbered from 0 to $m-1$. They are standing on a coordinate axis in points with positive integer coordinates. They are facing right (i.e. in the direction of the coordinate increase). At this moment everyone will start running with the constant speed in the direction of coordinate increasing. The initial coordinate of the i-th person on the line is x_i, and the speed of the i-th person is v_i. So the coordinate of the i-th person at the moment t will be $x_i + t \cdot v_i$.

Cirno captured the coordinates of m people in k consecutive integer moments from 0 to $k-1$. In every moment, the coordinates of m people were recorded in **arbitrary order**.

To make the problem more funny, Cirno modified one coordinate at the moment y ($0 < y < k-1$) to a **different** integer.

AquaMoon wants to find the moment y and the original coordinate p before the modification. Actually, she is not a programmer at all. So she wasn't able to solve it. Can you help her?

Input

This problem is made as interactive. It means, that your solution will read the input, given by the interactor. But the interactor will give you the full input at the beginning and after that, you should print the answer. So you should solve the problem, like as you solve the usual, non-interactive problem because you won't have any interaction

process. The only thing you should not forget is to flush the output buffer, after printing the answer. Otherwise, you can get an "Idleness limit exceeded" verdict. Refer to the interactive problems guide for the detailed information about flushing the output buffer.

The first line contains two integers m and k ($5 \le m \le 1000$, $7 \le k \le 1000$) — the number of people and the number of recorded moments.

The next k lines contain captured positions. i-th of these lines contains m integers between 1 and 10^6 (inclusive), representing positions captured by Cirno at the moment $i-1$.

The input is guaranteed to be valid (i.e. only one integer was modified to a different value according to the problem statement). Also, it is guaranteed, that $1 \le v_i \le 1000$ for all $1 \le i \le m$.

Hack format:

The first line should contain two integers m and k ($5 \le m \le 1000$, $7 \le k \le 1000$) — the number of people and the number of moments.

In the second line, there should be m integers $x_0, x_1, \dots, x_{m-1}$ ($1 \le x_i \le 10^6$), where x_i is the initial coordinate of the i-th person.

In the third line, there should be m integers $v_0, v_1, \dots, v_{m-1}$ ($1 \le v_i \le 1000$), where v_i is the speed of the i-th person. It should be true that $x_i + (k-1)v_i \le 10^6$ for each $0 \le i < m$.

In the next k lines, each line should contain m integers. i-th line should contain m distinct integers $p_0, p_1, \dots, p_{m-1}$ ($0 \le p_j < m$). The meaning of these numbers: j-th integer in the input in the i-th moment is the coordinate of the p_j-th person.

In the last line, there should be three integers y, i, c. Cirno modified the coordinate of the i-th person at the moment y to c ($1 \le y \le k-2$, $0 \le i \le m-1$, $1 \le c \le 10^6$, $c \neq x_i + y \cdot v_i$).

Output

Print a single line with two integers y, p — the moment that contains the modified coordinate and the original coordinate.

Example

input

```
5 7
6 9 9 6 9
10 7 10 8 10
11 11 11 10 8
12 12 12 12 9
14 13 12 10 13
11 14 16 14 14
12 15 18 15 15
```

output

```
4 13
```

Note

In the first test the initial coordinates of people are 9, 6, 6, 9, 9 and their speeds are 1, 2, 1, 1, 1. So, it's easy to see, that at the moment 4 one coordinate was modified from 13 to 12.

This is the first test in the hack format:

```
5 7
9 6 6 9 9
1 2 1 1 1
2 3 4 1 0
0 2 3 1 4
4 3 0 1 2
1 3 4 0 2
1 4 0 2 3
2 4 1 3 0
2 4 1 3 0
4 0 12
```

E1. AquaMoon and Time Stop (easy version)

time limit per test: 3 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Note that the differences between easy and hard versions are the constraints on n and the time limit. You can make hacks only if both versions are solved.

AquaMoon knew through foresight that some ghosts wanted to curse tourists on a pedestrian street. But unfortunately, this time, these ghosts were hiding in a barrier, and she couldn't enter this barrier in a short time and destroy them. Therefore, all that can be done is to save any unfortunate person on the street from the ghosts.

The pedestrian street can be represented as a one-dimensional coordinate system. There is one person hanging out on the pedestrian street. At the time 0 he is at coordinate x, moving with a speed of 1 unit per second. In particular, at time i the person will be at coordinate $x + i$.

The ghosts are going to cast n curses on the street. The i-th curse will last from time $tl_i - 1 + 10^{-18}$ to time $tr_i + 1 - 10^{-18}$ (exclusively) and will kill people with coordinates from $l_i - 1 + 10^{-18}$ to $r_i + 1 - 10^{-18}$ (exclusively). Formally that means, that the person, whose coordinate is between $(l_i - 1 + 10^{-18}, r_i + 1 - 10^{-18})$ in the time range $(tl_i - 1 + 10^{-18}, tr_i + 1 - 10^{-18})$ will die.

To save the person on the street, AquaMoon can stop time at any moment t, and then move the person from his current coordinate x to any coordinate y (t, x and y are not necessarily integers). The movement costs AquaMoon $|x - y|$ energy. The movement is continuous, so if there exists some cursed area between points x and y at time t, the person will **die too**.

AquaMoon wants to know what is the minimum amount of energy she needs to spend in order to save the person on the street from all n curses. But she is not good at programming. As her friend, can you help her?

Input

The first line contains a single integer n ($1 \le n \le 2000$) — the number of curses.

The next line contains a single integer x ($1 \le x \le 10^6$) — the initial coordinate of the person.

The following n lines contain four integers tl_i, tr_i, l_i, r_i each ($1 \le tl_i \le tr_i \le 10^6$, $1 \le l_i \le r_i \le 10^6$).

Output

Print a single integer — the minimum energy which AquaMoon needs to spent, rounded up to the nearest integer (in case there are two nearest integers you should round the answer to the highest of them).

Examples

input
```
2
1
1 2 1 2
2 3 2 3
```
output
```
2
```

input
```
3
4
1 4 1 2
1 4 4 15
6 7 1 4
```
output
```
8
```

input
```
4
3
1 5 1 1
4 10 1 4
1 2 3 13
1 10 7 19
```
output
```
14
```

input
```
7
5
78 96 76 91
6 16 18 37
53 63 40 56
83 88 21 38
72 75 17 24
63 63 53 60
34 46 60 60
```
output
```
20
```

E2. AquaMoon and Time Stop (hard version)

time limit per test: 7 seconds

memory limit per test: 1024 megabytes
input: standard input
output: standard output

Note that the differences between easy and hard versions are the constraints on n and the time limit. You can make hacks only if both versions are solved.

AquaMoon knew through foresight that some ghosts wanted to curse tourists on a pedestrian street. But unfortunately, this time, these ghosts were hiding in a barrier, and she couldn't enter this barrier in a short time and destroy them. Therefore, all that can be done is to save any unfortunate person on the street from the ghosts.

The pedestrian street can be represented as a one-dimensional coordinate system. There is one person hanging out on the pedestrian street. At the time 0 he is at coordinate x, moving with a speed of 1 unit per second. In particular, at time i the person will be at coordinate $x+i$.

The ghosts are going to cast n curses on the street. The i-th curse will last from time tl_i-1+10^{-18} to time tr_i+1-10^{-18} (exclusively) and will kill people with coordinates from l_i-1+10^{-18} to r_i+1-10^{-18} (exclusively). Formally that means, that the person, whose coordinate is between $(l_i-1+10^{-18},r_i+1-10^{-18})$ in the time range $(tl_i-1+10^{-18},tr_i+1-10^{-18})$ will die.

To save the person on the street, AquaMoon can stop time at any moment t, and then move the person from his current coordinate x to any coordinate y (t, x and y are not necessarily integers). The movement costs AquaMoon $|x-y|$ energy. The movement is continuous, so if there exists some cursed area between points x and y at time t, the person will **die too**.

AquaMoon wants to know what is the minimum amount of energy she needs to spend in order to save the person on the street from all n curses. But she is not good at programming. As her friend, can you help her?

Input

The first line contains a single integer n ($1 \le n \le 2\cdot 10^5$) — the number of curses.

The next line contains a single integer x ($1 \le x \le 10^6$) — the initial coordinate of the person.

The following n lines contain four integers tl_i, tr_i, l_i, r_i each ($1 \le tl_i \le tr_i \le 10^6$, $1 \le l_i \le r_i \le 10^6$).

Output

Print a single integer — the minimum energy which AquaMoon needs to spent, rounded up to the nearest integer (in case there are two nearest integers you should round the answer to the highest of them).

Examples

input

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

output

```
2
```

input

```
3
4
1 4 1 2
1 4 4 15
6 7 1 4
```

output

```
8
```

input

```
4
3
1 5 1 1
4 10 1 4
1 2 3 13
1 10 7 19
```

output

```
14
```

input

```
7
5
78 96 76 91
6 16 18 37
53 63 40 56
83 88 21 38
72 75 17 24
63 63 53 60
```

```
34 46 60 60
```

output

```
20
```

F. AquaMoon and Potatoes

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

AquaMoon has three integer arrays a, b, c of length n, where $1 \le a_i, b_i, c_i \le n$ for all i.

In order to accelerate her potato farming, she organizes her farm in a manner based on these three arrays. She is now going to complete m operations to count how many potatoes she can get. Each operation will have one of the two types:

1. AquaMoon reorganizes their farm and makes the k-th element of the array a equal to x. In other words, perform the assignment $a_k := x$.
2. Given a positive integer r, AquaMoon receives a potato for each triplet (i, j, k), such that $1 \le i < j < k \le r$, and $b_{a_i} = a_j = c_{a_k}$. Count the number of such triplets.

As AquaMoon is busy finding the library, help her complete all of their operations.

Input

The first line contains two integers n, m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 5 \cdot 10^4$).

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

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$).

The fourth line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le n$).

The next m lines describe operations, the i-th line describes the i-th operation in one of these two formats:

- "$1\ k\ x$" ($1 \le k, x \le n$), representing an operation of the first type.
- "$2\ r$" ($1 \le r \le n$), representing an operation of the second type.

It is guaranteed that there is at least one operation of the second type.

Output

For each operation of the second type print the answer.

Example

input

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

output

```
3
0
2
```

Note

For the first operation, the triplets are:

- $i = 1, j = 2, k = 3$
- $i = 2, j = 3, k = 4$
- $i = 3, j = 4, k = 5$

There is no satisfying triplet for the third operation.

For the fourth operation, the triplets are:

- $i = 2, j = 4, k = 5$
- $i = 3, j = 4, k = 5$