A. DIY Wooden Ladder

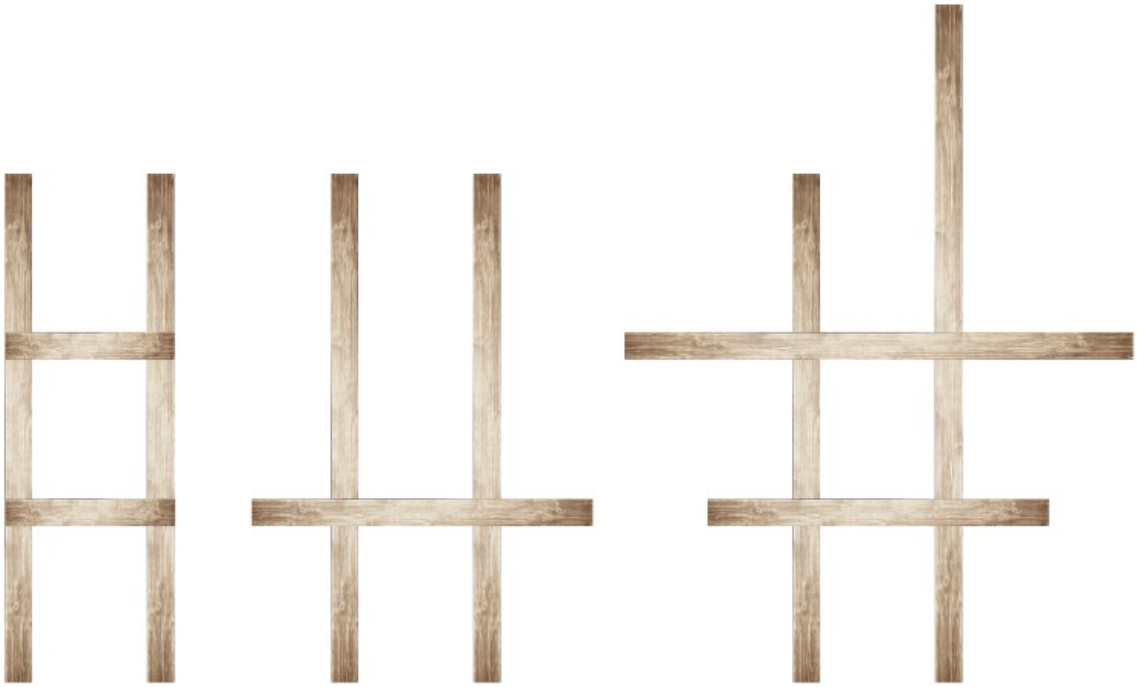
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's denote a k-step ladder as the following structure: exactly $k + 2$ wooden planks, of which

- two planks of length **at least** $k + 1$ — the base of the ladder;
- k planks of length **at least** 1 — the steps of the ladder;

Note that neither the base planks, nor the steps planks are required to be equal.

For example, ladders 1 and 3 are correct 2-step ladders and ladder 2 is a correct 1-step ladder. On the first picture the lengths of planks are $[3, 3]$ for the base and $[1]$ for the step. On the second picture lengths are $[3, 3]$ for the base and $[2]$ for the step. On the third picture lengths are $[3, 4]$ for the base and $[2, 3]$ for the steps.



You have n planks. The length of the i-th planks is a_i. You don't have a saw, so you can't cut the planks you have. Though you have a hammer and nails, so you can assemble the improvised "ladder" from the planks.

The question is: what is the maximum number k such that you can choose some subset of the given planks and assemble a k-step ladder using them?

Input

The first line contains a single integer T ($1 \le T \le 100$) — the number of queries. The queries are independent.

Each query consists of two lines. The first line contains a single integer n ($2 \le n \le 10^5$) — the number of planks you have.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$) — the lengths of the corresponding planks.

It's guaranteed that the total number of planks from all queries doesn't exceed 10^5.

Output

Print T integers — one per query. The i-th integer is the maximum number k, such that you can choose some subset of the planks given in the i-th query and assemble a k-step ladder using them.

Print 0 if you can't make even 1-step ladder from the given set of planks.

Example

input
4 4 1 3 1 3 3 3 3 2 5 2 3 3 4 2 3 1 1 2
output
2 1 2 0

Note

Examples for the queries $1 - 3$ are shown at the image in the legend section.

The Russian meme to express the quality of the ladders:

B. Pillars

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n pillars aligned in a row and numbered from 1 to n.

Initially each pillar contains exactly one disk. The i-th pillar contains a disk having radius a_i.

You can move these disks from one pillar to another. You can take a disk from pillar i and place it on top of pillar j if all these conditions are met:

1. there is no other pillar between pillars i and j. Formally, it means that $|i - j| = 1$;
2. pillar i contains **exactly** one disk;
3. either pillar j contains no disks, or the topmost disk on pillar j has radius strictly greater than the radius of the disk you move.

When you place a disk on a pillar that already has some disks on it, you put the new disk on top of previously placed disks, so the new disk will be used to check the third condition if you try to place another disk on the same pillar.

You may take any disk and place it on other pillar any number of times, provided that every time you do it, all three aforementioned conditions are met. Now you wonder, is it possible to place all n disks on the same pillar simultaneously?

Input

The first line contains one integer n ($3 \le n \le 2 \cdot 10^5$) — the number of pillars.

The second line contains n integers a_1, a_2, ..., a_i ($1 \le a_i \le n$), where a_i is the radius of the disk initially placed on the i-th pillar. All numbers a_i are distinct.

Output

Print YES if it is possible to place all the disks on the same pillar simultaneously, and NO otherwise. You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Examples

input
4 1 3 4 2
output
YES

input
3 3 1 2
output
NO

Note
In the first case it is possible to place all disks on pillar 3 using the following sequence of actions:

1. take the disk with radius 3 from pillar 2 and place it on top of pillar 3;
2. take the disk with radius 1 from pillar 1 and place it on top of pillar 2;
3. take the disk with radius 2 from pillar 4 and place it on top of pillar 3;
4. take the disk with radius 1 from pillar 2 and place it on top of pillar 3.

C. Array Splitting

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

You are given a **sorted** array $a_1, a_2, \ldots, a_n$ (for each index $i > 1$ condition $a_i \geq a_{i-1}$ holds) and an integer k.

You are asked to divide this array into k non-empty consecutive subarrays. Every element in the array should be included in exactly one subarray.

Let $max(i)$ be equal to the maximum in the i-th subarray, and $min(i)$ be equal to the minimum in the i-th subarray. The cost of division is equal to $\sum_{i=1}^{k}(max(i) - min(i))$. For example, if $a = [2, 4, 5, 5, 8, 11, 19]$ and we divide it into 3 subarrays in the following way: $[2, 4], [5, 5], [8, 11, 19]$, then the cost of division is equal to $(4 - 2) + (5 - 5) + (19 - 8) = 13$.

Calculate the minimum cost you can obtain by dividing the array a into k non-empty consecutive subarrays.

Input
The first line contains two integers n and k ($1 \leq k \leq n \leq 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$, $a_i \geq a_{i-1}$).

Output
Print the minimum cost you can obtain by dividing the array a into k nonempty consecutive subarrays.

Examples

input
6 3 4 8 15 16 23 42
output
12

input
4 4 1 3 3 7
output
0

input
8 1 1 1 2 3 5 8 13 21
output
20

Note
In the first test we can divide array a in the following way: $[4, 8, 15, 16], [23], [42]$.

D. Yet Another Subarray Problem

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

You are given an array $a_1, a_2, \ldots, a_n$ and two integers m and k.

You can choose some subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$.

The cost of subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$ is equal to $\sum\limits_{i=l}^{r} a_i - k\lceil \frac{r-l+1}{m} \rceil$, where $\lceil x \rceil$ is the least integer greater than or equal to x.

The cost of empty subarray is equal to zero.

For example, if $m = 3$, $k = 10$ and $a = [2, -4, 15, -3, 4, 8, 3]$, then the cost of some subarrays are:

- $a_3 \ldots a_3 : 15 - k\lceil \frac{1}{3} \rceil = 15 - 10 = 5$;
- $a_3 \ldots a_4 : (15 - 3) - k\lceil \frac{2}{3} \rceil = 12 - 10 = 2$;
- $a_3 \ldots a_5 : (15 - 3 + 4) - k\lceil \frac{3}{3} \rceil = 16 - 10 = 6$;
- $a_3 \ldots a_6 : (15 - 3 + 4 + 8) - k\lceil \frac{4}{3} \rceil = 24 - 20 = 4$;
- $a_3 \ldots a_7 : (15 - 3 + 4 + 8 + 3) - k\lceil \frac{5}{3} \rceil = 27 - 20 = 7$.

Your task is to find the maximum cost of some subarray (possibly empty) of array a.

Input
The first line contains three integers n, m, and k ($1 \le n \le 3 \cdot 10^5, 1 \le m \le 10, 1 \le k \le 10^9$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output
Print the maximum cost of some subarray of array a.

Examples

input
7 3 10 2 -4 15 -3 4 8 3
output
7

input
5 2 1000 -13 -4 -9 -20 -11
output
0

E. Culture Code

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

There are famous Russian nesting dolls named matryoshkas sold in one of the souvenir stores nearby, and you'd like to buy several of them. The store has n different matryoshkas. Any matryoshka is a figure of volume out_i with an empty space inside of volume in_i (of course, $out_i > in_i$).

You don't have much free space inside your bag, but, fortunately, you know that matryoshkas can be nested one inside another. Formally, let's call a set of matryoshkas *nested* if we can rearrange dolls in such a way, that the first doll can be nested inside the second one, the second doll — inside the third one and so on. Matryoshka i can be nested inside matryoshka j if $out_i \le in_j$. So only the last doll will take space inside your bag.

Let's call *extra space* of a nested set of dolls as a total volume of empty space inside this structure. Obviously, it's equal to $in_{i_1} + (in_{i_2} - out_{i_1}) + (in_{i_3} - out_{i_2}) + \cdots + (in_{i_k} - out_{i_{k-1}})$, where i_1, i_2, ..., i_k are the indices of the chosen dolls in the order they are nested in each other.

Finally, let's call a nested subset of the given sequence as *big enough* if there isn't any doll from the sequence that can be added to the nested subset without breaking its nested property.

You want to buy many matryoshkas, so you should choose a *big enough* nested subset to buy it. But you will be disappointed if too much space in your bag will be wasted, so you want to choose a big enough subset so that its *extra space* is minimum possible among all big enough subsets. Now you wonder, how many different nested subsets meet these conditions (they are big enough, and there is no big enough subset such that its extra space is less than the extra space of the chosen subset). Two subsets are considered different if there exists at least one index i such that one of the subsets contains the i-th doll, and another subset doesn't.

Since the answer can be large, print it modulo $10^9 + 7$.

Input

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

The next n lines contain a description of each doll: two integers out_i and in_i ($1 \le in_i < out_i \le 10^9$) — the outer and inners volumes of the i-th matryoshka.

Output
Print one integer — the number of big enough nested subsets such that extra space of each of these subsets is minimum possible. Since the answer can be large, print it modulo $10^9 + 7$.

Example

input
7 4 1 4 2 4 2 2 1 5 4 6 4 3 2
output
6

Note
There are 6 big enough nested subsets with minimum possible extra space in the example:

- $\{1, 5\}$: we can't add any other matryoshka and keep it nested; it's extra space is 1;
- $\{1, 6\}$;
- $\{2, 4, 5\}$;
- $\{2, 4, 6\}$;
- $\{3, 4, 5\}$;
- $\{3, 4, 6\}$.

There are no more "good" subsets because, for example, subset $\{6, 7\}$ is not big enough (we can add the 4-th matryoshka to it) or subset $\{4, 6, 7\}$ has extra space equal to 2.

F. Coloring Game

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alice and Bob want to play a game. They have n colored paper strips; the i-th strip is divided into a_i cells numbered from 1 to a_i. Each cell can have one of 3 colors.

In the beginning of the game, Alice and Bob put n chips, the i-th chip is put in the a_i-th cell of the i-th strip. Then they take turns, Alice is first. Each player during their turn has to choose one chip and move it 1, 2 or 3 cells backwards (i. e. if the current cell is x, then the chip can be moved to the cell $x - 1$, $x - 2$ or $x - 3$). There are two restrictions: the chip cannot leave the borders of the strip (for example, if the current cell is 3, then you can't move the chip 3 cells backwards); and some moves may be prohibited because of color of the current cell (a matrix f with size 3×3 is given, where $f_{i,j} = 1$ if it is possible to move the chip j cells backwards from the cell which has color i, or $f_{i,j} = 0$ if such move is prohibited). The player who cannot make a move loses the game.

Initially some cells may be uncolored. Bob can color all uncolored cells as he wants (but he cannot leave any cell uncolored). Let's call a coloring *good* if Bob can win the game no matter how Alice acts, if the cells are colored according to this coloring. Two colorings are different if at least one cell is colored in different colors in these two colorings.

Bob wants you to calculate the number of good colorings. Can you do it for him?

Since the answer can be really large, you have to print it modulo 998244353.

Input
The first line contains one integer n — the number of paper strips ($1 \le n \le 1000$).

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$), where a_i is the number of cells in the i-th strip.

The third line contains one integer m ($1 \le m \le 1000$) — the number of cells that are already colored.

Then m lines follow, each describing an already colored cell. The i-th of these lines contains three integers x_i, y_i and c_i ($1 \le x_i \le n$, $1 \le y_i \le a_{x_i}$, $1 \le c_i \le 3$) denoting that the cell y_i in the strip x_i has color c_i. It is guaranteed that if $i \ne j$, then either $x_i \ne x_j$ or $y_i \ne y_j$ (or both).

Then 3 lines follow, i-th line containing 3 numbers $f_{i,1}$, $f_{i,2}$, $f_{i,3}$ ($0 \le f_{i,j} \le 1$). If $f_{i,j} = 1$, then it is possible to move the chip j cells backwards from the cell having color i; if $f_{i,j} = 0$, then such move is impossible.

Output

Print one integer: the number of good colorings, taken modulo 998244353.

Examples

input
3 3 4 5 2 1 1 1 2 2 2 1 1 1 1 0 0 0 1 1
output
14346

input
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
output
1

input
3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
output
9