Technocup 2022 - Elimination Round 3

A. Life of a Flower

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

Petya has got an interesting flower. Petya is a busy person, so he sometimes forgets to water it. You are given n days from Petya's live and you have to determine what happened with his flower in the end.

The flower grows as follows:

- If the flower isn't watered for two days in a row, it dies.
- If the flower is watered in the i-th day, it grows by 1 centimeter.
- If the flower is watered in the i-th and in the $(i-1)$-th day ($i > 1$), then it grows by 5 centimeters instead of 1.
- If the flower is not watered in the i-th day, it does not grow.

At the beginning of the 1-st day the flower is 1 centimeter tall. What is its height after n days?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains the only integer n ($1 \le n \le 100$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($a_i = 0$ or $a_i = 1$). If $a_i = 1$, the flower is watered in the i-th day, otherwise it is not watered.

Output

For each test case print a single integer k — the flower's height after n days, or -1, if the flower dies.

Example

input
4 3 1 0 1 3 0 1 1 4 1 0 0 1 1 0
output
3 7 -1 1

B. Array Eversion

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

You are given an array a of length n.

Let's define the *eversion* operation. Let $x = a_n$. Then array a is partitioned into two parts: left and right. The left part contains the elements of a that are not greater than x ($\le x$). The right part contains the elements of a that are strictly greater than x ($> x$). The order of elements in each part is kept the same as before the operation, i. e. the partition is stable. Then the array is replaced with the concatenation of the left and the right parts.

For example, if the array a is $[2, 4, 1, 5, 3]$, the eversion goes like this: $[2, 4, 1, 5, 3] \to [2, 1, 3], [4, 5] \to [2, 1, 3, 4, 5]$.

We start with the array a and perform eversions on this array. We can prove that after several eversions the array a stops changing. Output the minimum number k such that the array stops changing after k eversions.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases

follows.

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

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

Output
For each test case print a single integer k — the number of eversions after which the array stops changing.

Example

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

Note
Consider the fist example.

- The first eversion: $a = [1, 4, 2, 5, 3]$, $x = 3$. $[2, 4, 1, 5, 3] \to [2, 1, 3], [4, 5] \to [2, 1, 3, 4, 5]$.
- The second and following eversions: $a = [2, 1, 3, 4, 5]$, $x = 5$. $[2, 1, 3, 4, 5] \to [2, 1, 3, 4, 5], [] \to [2, 1, 3, 4, 5]$. This eversion does not change the array, so the answer is 1.

Consider the second example.

- The first eversion: $a = [5, 3, 2, 4, 1]$, $x = 1$. $[5, 3, 2, 4, 1] \to [1], [5, 3, 2, 4] \to [1, 5, 3, 2, 4]$.
- The second eversion: $a = [1, 5, 3, 2, 4]$, $x = 4$. $[1, 5, 3, 2, 4] \to [1, 3, 2, 4], [5] \to [1, 3, 2, 4, 5]$.
- The third and following eversions: $a = [1, 3, 2, 4, 5]$, $x = 5$. $[1, 3, 2, 4, 5] \to [1, 3, 2, 4, 5], [] \to [1, 3, 2, 4, 5]$. This eversion does not change the array, so the answer is 2.

C. Minimize Distance

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

A total of n depots are located on a number line. Depot i lies at the point x_i for $1 \le i \le n$.

You are a salesman with n bags of goods, attempting to deliver one bag to each of the n depots. You and the n bags are initially at the origin 0. You can carry up to k bags at a time. You must collect the required number of goods from the origin, deliver them to the respective depots, and then return to the origin to collect your next batch of goods.

Calculate the minimum distance you need to cover to deliver all the bags of goods to the depots. You do **not** have to return to the origin after you have delivered all the bags.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,500$). Description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$).

The second line of each test case contains n integers $x_1, x_2, \ldots, x_n$ ($-10^9 \le x_i \le 10^9$). It is possible that some depots share the same position.

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

Output
For each test case, output a single integer denoting the minimum distance you need to cover to deliver all the bags of goods to the depots.

Example

input
4 5 1 1 2 3 4 5 9 3 -5 -10 -15 6 5 8 3 7 4

5 3
2 2 3 3 3
4 2
1000000000 1000000000 1000000000 1000000000

output

25
41
7
3000000000

Note

In the first test case, you can carry only one bag at a time. Thus, the following is a solution sequence that gives a minimum travel distance: $0 \to 2 \to 0 \to 4 \to 0 \to 3 \to 0 \to 1 \to 0 \to 5$, where each 0 means you go the origin and grab one bag, and each positive integer means you deliver the bag to a depot at this coordinate, giving a total distance of 25 units. It must be noted that there are other sequences that give the same distance.

In the second test case, you can follow the following sequence, among multiple such sequences, to travel minimum distance: $0 \to 6 \to 8 \to 7 \to 0 \to 5 \to 4 \to 3 \to 0 \to (-5) \to (-10) \to (-15)$, with distance 41. It can be shown that 41 is the optimal distance for this test case.

D. Yet Another Sorting Problem

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

Petya has an array of integers $a_1, a_2, \ldots, a_n$. He only likes sorted arrays. Unfortunately, the given array could be arbitrary, so Petya wants to sort it.

Petya likes to challenge himself, so he wants to sort array using only 3-cycles. More formally, in one operation he can pick 3 **pairwise distinct** indices i, j, and k ($1 \le i, j, k \le n$) and apply $i \to j \to k \to i$ cycle to the array a. It simultaneously places a_i on position j, a_j on position k, and a_k on position i, without changing any other element.

For example, if a is $[10, 50, 20, 30, 40, 60]$ and he chooses $i = 2$, $j = 1$, $k = 5$, then the array becomes $[\underline{50}, \underline{40}, 20, 30, \underline{10}, 60]$.

Petya can apply arbitrary number of 3-cycles (possibly, zero). You are to determine if Petya can sort his array a, i. e. make it non-decreasing.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^5$). Description of the test cases follows.

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

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

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

Output

For each test case, print "YES" (without quotes) if Petya can sort the array a using 3-cycles, and "NO" (without quotes) otherwise. You can print each letter in any case (upper or lower).

Example

input

7
1
1
2
2 2
2
2 1
3
1 2 3
3
2 1 3
3
3 1 2
4
2 1 4 3

output

YES
YES
NO
YES
NO
YES
YES

Note
In the 6-th test case Petya can use the 3-cycle $1 \to 3 \to 2 \to 1$ to sort the array.

In the 7-th test case Petya can apply $1 \to 3 \to 2 \to 1$ and make $a = [1, 4, 2, 3]$. Then he can apply $2 \to 4 \to 3 \to 2$ and finally sort the array.

E. Frequency Queries

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Petya has a rooted tree with an integer written on each vertex. The vertex 1 is the root. You are to answer some questions about the tree.

A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a node v is the next vertex on the shortest path from v to the root.

Each question is defined by three integers v, l, and k. To get the answer to the question, you need to perform the following steps:

- First, write down the sequence of all integers written on the shortest path from the vertex v to the root (including those written in the v and the root).
- Count the number of times each integer occurs. Remove all integers with less than l occurrences.
- Replace the sequence, removing all duplicates and ordering the elements by the number of occurrences in the original list in increasing order. In case of a tie, you can choose the order of these elements arbitrary.
- The answer to the question is the k-th number in the remaining sequence. Note that the answer is not always uniquely determined, because there could be several orderings. Also, it is possible that the length of the sequence on this step is less than k, in this case the answer is -1.

For example, if the sequence of integers on the path from v to the root is $[2, 2, 1, 7, 1, 1, 4, 4, 4]$, $l = 2$ and $k = 2$, then the answer is 1.

Please answer all questions about the tree.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^6$). Description of the test cases follows.

The first line of each test case contains two integers n, q ($1 \le n, q \le 10^6$) — the number of vertices in the tree and the number of questions.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), where a_i is the number written on the i-th vertex.

The third line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i \le n$), where p_i is the parent of node i. It's guaranteed that the values p define a correct tree.

Each of the next q lines contains three integers v, l, k ($1 \le v, l, k \le n$) — descriptions of questions.

It is guaranteed that the sum of n and the sum of q over all test cases do not exceed 10^6.

Output
For each question of each test case print the answer to the question. In case of multiple answers, print any.

Example

input
2 3 3 1 1 1 1 2 3 1 1 3 1 2 3 2 1 5 5 1 2 1 1 2 1 1 2 2 3 1 1 2 1 2 4 1 1 4 2 1 4 2 2
output
1 -1 1 1 1 2 1 -1

F. Non-equal Neighbours

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array of n positive integers $a_1, a_2, \ldots, a_n$. Your task is to calculate the number of arrays of n positive integers $b_1, b_2, \ldots, b_n$ such that:

- $1 \le b_i \le a_i$ for every i ($1 \le i \le n$), and
- $b_i \neq b_{i+1}$ for every i ($1 \le i \le n - 1$).

The number of such arrays can be very large, so print it modulo $998\,244\,353$.

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output
Print the answer modulo $998\,244\,353$ in a single line.

Examples

input
3 2 2 2
output
2

input
2 2 3
output
4

input
3 1 1 1
output
0

Note
In the first test case possible arrays are $[1, 2, 1]$ and $[2, 1, 2]$.

In the second test case possible arrays are $[1, 2]$, $[1, 3]$, $[2, 1]$ and $[2, 3]$.

G. Poachers

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

Alice and Bob are two poachers who cut trees in a forest.

A forest is a set of zero or more trees. A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a node v is the next vertex on the shortest path from v to the root. Children of vertex v are all nodes for which v is the parent. A vertex is a leaf if it has no children.

In this problem we define the *depth* of vertex as number of vertices on the simple path from this vertex to the root. The *rank* of a tree is the minimum depth among its leaves.

Initially there is a forest of rooted trees. Alice and Bob play a game on this forest. They play alternating turns with Alice going first. At the beginning of their turn, the player chooses a tree from the forest. Then the player chooses a positive *cutting depth*, which should **not exceed the rank** of the chosen tree. Then the player removes all vertices of that tree whose depth is less that or equal to the cutting depth. All other vertices of the tree form a set of rooted trees with root being the vertex with the smallest depth before the cut. All these trees are included in the game forest and the game continues.

A player loses if the forest is empty at the beginning of his move.

You are to determine whether Alice wins the game if both players play optimally.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^5$). Description of the test

cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^5$) — total number of vertices in the initial forest.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le n$) — description of the forest. If $p_i = 0$, then the i-th vertex is the root of a tree, otherwise p_i is the parent of the vertex i. It's guaranteed that p defines a correct forest of rooted trees.

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

Output

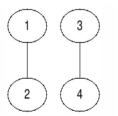
For each test case, print "YES" (without quotes) if Alice wins, otherwise print "NO" (without quotes). You can print each letter in any case (upper or lower).

Example

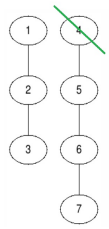
input
4 4 0 1 0 3 7 0 1 2 0 4 5 6 4 0 1 1 2 7 0 1 1 2 2 3 3
output
NO YES NO YES

Note

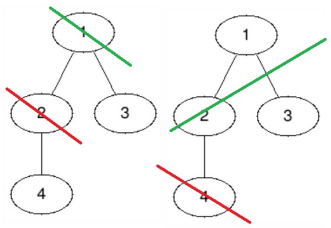
In the first test case Bob has a symmetric strategy, so Alice cannot win.



In the second test case Alice can choose the second tree and cutting depth 1 to get a forest on which she has a symmetric strategy.



In third test case the rank of the only tree is 2 and both possible moves for Alice result in a loss. Bob either can make the forest with a symmetric strategy for himself, or clear the forest.



In the fourth test case all leafs have the same depth, so Alice can clear the forest in one move.