F1. Flying Sort (Easy Version)

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

This is an easy version of the problem. In this version, all numbers in the given array are distinct and the constraints on n are less than in the hard version of the problem.

You are given an array a of n integers **(there are no equals elements in the array)**. You can perform the following operations on array elements:

1. choose any index i $(1 \le i \le n)$ and move the element $a[i]$ to the **begin** of the array;
2. choose any index i $(1 \le i \le n)$ and move the element $a[i]$ to the **end** of the array.

For example, if $n = 5$, $a = [4, 7, 2, 3, 9]$, then the following sequence of operations can be performed:

- after performing the operation of the first type to the second element, the array a will become $[7, 4, 2, 3, 9]$;
- after performing the operation of the second type to the second element, the array a will become $[7, 2, 3, 9, 4]$.

You can perform operations of any type any number of times in any order.

Find the minimum total number of operations of the first and second type that will make the a array sorted in non-decreasing order. In other words, what is the minimum number of operations that must be performed so the array satisfies the inequalities $a[1] \le a[2] \le \ldots \le a[n]$.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases in the test. Then t test cases follow.

Each test case starts with a line containing an integer n $(1 \le n \le 3000)$ — length of the array a.

Then follow n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$ — an array that needs to be sorted by the given operations. **All numbers in the given array are distinct**.

The sum of n for all test cases in one test does not exceed 3000.

Output

For each test case output one integer — the minimum total number of operations of the first and second type, which will make the array sorted in non-decreasing order.

Example

input Copy

```
4
5
4 7 2 3 9
5
3 5 8 1 7
5
1 4 5 7 12
4
0 2 1 3
```

output Copy

```
2
2
0
2
```

Note

In the first test case, you first need to move 3, and then 2 to the beginning of the array. Therefore, the desired sequence of operations:
$[4, 7, 2, 3, 9] \rightarrow [3, 4, 7, 2, 9] \rightarrow [2, 3, 4, 7, 9]$.

In the second test case, you need to move the 1 to the beginning of the array, and the 8 — to the end. Therefore, the desired sequence of operations:
$[3, 5, 8, 1, 7] \rightarrow [1, 3, 5, 8, 7] \rightarrow [1, 3, 5, 7, 8]$.

In the third test case, the array is already sorted.