Dolls

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

A little chicken was playing a ring-toss game and ended up tossing rings onto sets of nesting dolls.

—Steamed-chicken

David has just obtained n Russian nesting dolls of distinct sizes. He arranges these dolls in a row from left to right, where the i-th position contains a doll of size a_i.

Let the size of the smallest doll in the i-th position be l_i, and the size of the largest one be r_i. Dolls over two adjacent positions i and $i+1$ can be merged if and only if $r_i < l_{i+1}$ or $r_{i+1} < l_i$. The new nesting doll will contain all the dolls from the original i-th and $i+1$-th positions and will be placed in the i-th position. All dolls in positions greater than $i+1$ will shift left by one position to fill the gap.

For example, when $n = 4, a = [2, 1, 4, 3]$, David can:

1. Merge the dolls in positions 1 and 2. Now the remaining dolls have sizes $[(1, 2), (4), (3)]$.
2. Merge the dolls in positions 2 and 3. Now the remaining dolls have sizes $[(1, 2), (3, 4)]$.
3. Merge the dolls in positions 1 and 2. Now all the dolls have been merged into one position.

How many merge operations at most can David perform under an optimal strategy?

Input

Each test file contains multiple test cases. The first line contains the number of test cases T ($1 \le T \le 10^4$). The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 10^5$), representing the number of nesting dolls.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$, $\forall i \ne j, a_i \ne a_j$), representing the initial sizes of the dolls in each position.

For each test file, it is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output a single integer on a new line, representing the maximum number of merge operations that can be performed.

Example

standard input	standard output
8 4 2 1 4 3 4 1 4 2 3 4 3 1 4 2 5 1 3 5 2 4 5 1 4 2 5 3 5 2 5 3 1 4 6 1 3 6 5 2 4 6 2 5 1 3 6 4	3 3 2 3 3 3 4 4