Task 2: `PANCAKE`

You are given a stack of N pancakes, where $1 \leq N \leq 8$. The pancake at the bottom and at the top of the stack has index 0 and index $N-1$ respectively. The size of a pancake is its diameter which is a positive integer. All pancakes in the stack have different sizes. A stack `A` of $N = 5$ pancakes with size 3, 8, 7, 6, and 10 can be visualized as:

```
4  (top)        10
3                6
2                7
1                8
0  (bottom)      3
-----------------------
index i       A[i]
```

We want to sort the stack in *descending order*, that is, the largest pancake is at the bottom and the smallest pancake is at the top. To make the problem more real-life like, sorting a stack of pancakes can only be done by a sequence of pancake 'flips', denoted by *flip*(i). The *flip*(i) operation inserts a spatula into the stack, lifts up pancakes from index i to index $N-1$ and flips them. As a result, the position of the pancakes from index i to $N-1$ are reversed.

For example, stack `A` can be transformed to stack `B` via *flip*(0), i.e. inserting a spatula and flipping the pancakes from index 0 and 4. Stack `B` can be transformed to stack `C` via *flip*(3). Stack `C` can be transformed to stack `D` via *flip*(1), and so on. Our target is to make the stack sorted in descending order, i.e. we want the stack to be like stack `E`, using minimum number of flips.

```
4  (top)       10  <--   3  <--   8  <--   6                 3
3               6        8  <--   3        7      ...        6
2               7        7        7        3                 7
1               8        6        6  <--   8                 8
0  (bottom)     3  <--  10       10       10                10
--------------------------------------------------------------
index i    A[i]     B[i]     C[i]     D[i]      ...      E[i]
```

Bill Gates (Microsoft founder, former CEO, and current chairman) wrote only one research paper so far, and it is about this pancake sorting[1].

In this question, given the starting configuration of N pancakes, your task is to compute the *minimum* number of flips required to sort them.

Input format

The first line of input contains an integer T which is the number of test cases. Each of the next T lines corresponds to a test case. Each test case starts with the number of pancakes N, and then followed by a sequence of N integers indicating the size of each pancake. The bottom pancake (i.e the pancake with index 0) appears first in the sequence. Examples will be given in the next page.

Output format

For each test case, output in one line the minimum number of flips required to sort the test case. Hence, the output contains T lines with one integer per line. Examples will be given in the next page.

Input instances

Your program will be tested on 5 sets of input instances as follow:

1. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 1 flip. The pancakes sizes range from 1 to N and we have $1 \leq T \leq 200$.

2. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 2 flips. The pancakes sizes range from 1 to N and we have $1 \leq T \leq 200$.

3. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 2 flips. The pancakes sizes range from 1 to $1,000,000$ and we have $1 \leq T \leq 200$.

4. (8 marks) Here, the minimum number of flips is not constrained, and thus could be large. The pancakes sizes range from 1 to $1,000,000$, and we have $1 \leq T \leq 200$.

5. (5 marks) Same as the fourth criteria above, but with $1 \leq T \leq 6000$. Since the number of test cases can be large, your program must be (very) efficient.

[1]Gates, W. and Papadimitriou, C. *Bounds for Sorting by Prefix Reversal,* Discrete Mathematics, 27, 47-57, 1979.

More Samples

Here are examples for each of the 5 input sets.

Example for set 1

Input

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

Output

```
0
1
```

Remark. The first test case is already sorted in descending order, so no flip is needed. The second test case can be sorted in descending order in 1 flip: *flip*(5).

Example for set 2

Input

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

Output

```
0
1
2
```

Remark. The first and the second test cases are exactly the same as the example for set 1. The third test case can be sorted in descending order in 2 flips: *flip*(3) then *flip*(1).

Example for set 3

Input

```
2
5 555555 111111 222222 444444 333333
8 1000000 999999 999998 999997 999996 999995 999994 999993
```

Output

```
2
0
```

Remark. The first input stack can be sorted in descending order by 2 flips: *flip*(3) then *flip*(1). The second input stack is already sorted in descending order, so no flip is needed.

Example for set 4 & 5

Input

```
3
5 555555 111111 222222 444444 333333
8 1000000 999999 999998 999997 999996 999995 999994 999993
5 3 8 7 6 10
```

Output

```
2
0
4
```

Remark. The first and the second test cases are exactly the same as the example for set 3. The third test case (which is the example given in the task description) can be sorted in 4 flips by the following two possible sequences:

- Solution 1: *flip*(0), *flip*(1), *flip*(2), *flip*(1): 4 flips.
- Solution 2: *flip*(1), *flip*(2), *flip*(1), *flip*(0): also 4 flips.

There is no way to sort using less than 4 flips, and thus 4 is the minimum possible.