UCF "Practice" Local Contest — August 19-23, 2024

First Last Sorting

filename: `firstlast`
Difficulty Level: `Medium-Hard`
Time Limit: `2 seconds`

Arup has just created a data structure that makes the two following list transformations in constant O(1) time:

a. Take any element in the list and move it to the front.
b. Take any element in the list and move it to the back.

You've realized that sorting speed can be improved using these transformations. For example, consider the input list:

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

We can do the following sequence of transformations to sort this list:

8,	3,	7,	4,	1,	5,	2,	6	(move 6 to end)
8,	3,	4,	1,	5,	2,	6,	7	(move 7 to end)
2,	8,	3,	4,	1,	5,	6,	7	(move 2 to front)
1,	2,	8,	3,	4,	5,	6,	7	(move 1 to front)
1,	2,	3,	4,	5,	6,	7,	8	(move 8 to end)

You are now curious. Given an input array of distinct values, what is the fewest number of these first/last operations necessary to sort the array?

The Problem:

Given an initial permutation of the integers 1, 2, ..., n, determine the fewest number of first/last operations necessary to get the list of values sorted in increasing order.

The Input:

The first line of input will contain a single positive integer, n ($n \leq 10^5$), representing the number of values to be sorted. The next n lines contain one integer each. All of these integers will be distinct values in between 1 and n (inclusive), representing the original order of the data to sort for the input case.

The Output:

On a line by itself, output the fewest number of first/last operations necessary to sort the input list.

Sample Input	Sample Output
8 8 3 6 7 4 1 5 2	5
5 1 2 5 3 4	1