Sorting Proximity (proximity)

As a New Year's resolution, our heroes want everything around them to be clean and perfectly arranged. Thus, they now want to apply their new ideals on their daily work, namely handling the various data structures they are using.

Figure 1: Some of the New Year's resolutions of our heroes.

This time around, they got an array V_i of length N and they define the sorting proximity of the array as the number of swaps which have to be done in order to sort the array, if we use the bubble sort algorithm.

Namely, as long as the array is not sorted yet, the bubble sort algorithm iterates through the array and every time two adjacent values are wrongly placed relative to one another, they are swapped. The sorting proximity is the number of times this happens during the algorithm. For example, if we have the array $[4, 2, 3, 5, 1]$, the sorting proximity of the array is 6, corresponding to the following 6 swaps:

- $[2, 4, 3, 5, 1]$
- $[2, 3, 4, 5, 1]$
- $[2, 3, 4, 1, 5]$
- $[2, 3, 1, 4, 5]$
- $[2, 1, 3, 4, 5]$
- $[1, 2, 3, 4, 5]$

Now, our heroes want to improve the sorting proximity of the array and they asked for your help. Your job is to swap two integers from the array (they don't have to be adjacent) so that the sorting proximity of the resulting array is minimal. Remember: even though the swap may not improve the answer, you must do it!

☞ Among the attachments of this task you may find a template file `proximity.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers V_i.

Output

You need to write a single line with an integer: the minimal sorting proximity after a single swap.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq V_i \leq 10^9$ for each $i = 0 \ldots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (35 points)	$N \leq 1000$ and the values from the array are pairwise distinct.
– **Subtask 3** (40 points)	All the values from the array are pairwise distinct.
– **Subtask 4** (25 points)	No additional limitations.

Examples

input	output

`5`
`5 2 4 3 1`
	`1`

`5`
`3 1 7 9 5`
	`2`

Explanation

In the **first sample case**, we can swap 5 and 1 and we will get the array $[1, 2, 4, 3, 5]$ which has the sorting proximity equal to 1.

In the **second sample case**, we can swap 5 and 7 and we will get the array $[3, 1, 5, 9, 7]$ which has the sorting proximity equal to 2.