bluewolf
an IBM Company

Developer I v2

27m : 39s
to section end

☆ Custom-Sorted Array

- Coding Section -

1

2

- Test -

3

4

5

6

7

8

9

10

11

12

- Extra Credit -

13

In an array, *a*, of *n* positive integers, we can swap the elements at any two indices in a single operation called a *move*. For example, if our array is *a = [17, 4, 8]*, we can swap $a_0 = 17$ and $a_2 = 8$ to get *a = [8, 4, 17]* in a single move.

We want to custom-sort array *a* such that all of the *even* elements are at the beginning of the array and all of the *odd* elements are at the end of the array. For example, if our array is *a = [6, 3, 4, 5]*, then the following four arrays are valid custom-sorted arrays:

- *a = [6, 4, 3, 5]*
- *a = [4, 6, 3, 5]*
- *a = [6, 4, 5, 3]*
- *a = [4, 6, 5, 3]*

Complete the *moves* function in the editor below. It has one parameter: an array, *a*, of *n* positive integers. It must return an integer denoting the minimum number of moves required to sort the array's elements such that the reordered array contains all the *even* elements at the beginning of the array followed by all the *odd* elements at the end of the array. Note that the only ordering requirement for the custom-sorted array is that no odd element appears before any even element in the array, so the order of even elements with respect to other even elements and odd elements with respect to other odd elements does not matter.

Input Format

Locked stub code in the editor reads the following input from stdin and passes it to the function:
The first line contains an integer, *n*, denoting the number of elements in array *a*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer describing a_i.

Constraints

- $2 \le n \le 10^5$
- $1 \le a_i \le 10^9$, where $0 \le i < n$.
- It is guaranteed that array *a* contains at least one *even* and one *odd* element.

Output Format

The function must return an integer denoting the minimum number of moves required to custom-sort the array such that all the *even* elements are at the beginning and all the *odd* elements are at the end. This is printed to stdout by locked stub code in the editor.

bluewolf
an IBM Company

Developer I v2

27m : 39s
to section end

```
13
10
21
20
```

- Coding Section -

Sample Output 0

```
1
```

- Test -

Explanation 0

Given $a = [13, 10, 21, 20]$, we can swap a_0 and a_3 to get the custom-sorted array $a = [20, 10, 21, 13]$. It took one move to custom-sort array a and this value is minimal, so the function returns 1.

Sample Input 1

```
5
8
5
11
4
6
```

Sample Output 1

```
2
```

Explanation 1

We can perform the following moves on our initial array, $a = [8, 5, 11, 4, 6]$:

1. Swap a_1 and a_3 to get the array $a = [8, 4, 11, 5, 6]$.
2. Swap a_2 and a_4 to get the array $a = [8, 4, 6, 5, 11]$.

It took two moves to get a valid custom-sorted array. As this value is minimal, the function returns 2.

- Extra Credit -

YOUR ANSWER

bluewolf
an IBM Company

Developer I v2

27m : 39s
to section end

- Coding Section -

1

2

- Test -

3

4

5

6

7

8

9

10

11

12

- Extra Credit -

13

Draft saved 02:03 pm | Original code | Swift

```swift
import Foundation

/*
 * Complete the function below.
 */
func moves(a: [Int]) -> Int {


}



var _a_cnt = 0
_a_cnt = Int(readLine()!)!

var _a = [Int](repeating: 0, count: _a_cnt)

for _a_i in 0..<_a_cnt {
    var _a_item = 0
    if let _a_item_temp = readLine() {
        _a_item = Int(_a_item_temp)!
    }
    _a[_a_i] = _a_item
}

var res = moves(a: _a)

print(res)

```

Line: 11 Col: 1

bluewolf
an IBM Company

Developer I v2

27m : 39s
to section end

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

- Coding Section -

1

2

- Test -

3

4

5

6

7

8

9

10

11

12

- Extra Credit -

13