

Problem 2459: Sort Array by Moving Items to Empty Space

Problem Information

Difficulty: Hard

Acceptance Rate: 45.22%

Paid Only: Yes

Tags: Array, Greedy, Sorting

Problem Description

You are given an integer array `nums`` of size `n`` containing **each** element from `0`` to `n - 1`` (**inclusive**). Each of the elements from `1`` to `n - 1`` represents an item, and the element `0`` represents an empty space.

In one operation, you can move **any** item to the empty space. `nums`` is considered to be sorted if the numbers of all the items are in **ascending** order and the empty space is either at the beginning or at the end of the array.

For example, if `n = 4``, `nums`` is sorted if:

* `nums = [0,1,2,3]` or * `nums = [1,2,3,0]`

...and considered to be unsorted otherwise.

Return the **minimum** number of operations needed to sort `nums``.

Example 1:

Input: `nums = [4,2,0,3,1]` **Output:** 3 **Explanation:** - Move item 2 to the empty space. Now, `nums = [4,0,2,3,1]`. - Move item 1 to the empty space. Now, `nums = [4,1,2,3,0]`. - Move item 4 to the empty space. Now, `nums = [0,1,2,3,4]`. It can be proven that 3 is the minimum number of operations needed.

Example 2:

****Input:**** nums = [1,2,3,4,0] ****Output:**** 0 ****Explanation:**** nums is already sorted so return 0.

****Example 3:****

****Input:**** nums = [1,0,2,4,3] ****Output:**** 2 ****Explanation:**** - Move item 2 to the empty space. Now, nums = [1,2,0,4,3]. - Move item 3 to the empty space. Now, nums = [1,2,3,4,0]. It can be proven that 2 is the minimum number of operations needed.

****Constraints:****

* `n == nums.length` * `2 <= n <= 105` * `0 <= nums[i] < n` * All the values of `nums` are ****unique****.

Code Snippets

C++:

```
class Solution {
public:
    int sortArray(vector<int>& nums) {

    }

};
```

Java:

```
class Solution {
    public int sortArray(int[] nums) {

    }

}
```

Python3:

```
class Solution:
    def sortArray(self, nums: List[int]) -> int:
```