581. Shortest Unsorted Continuous Subarray

Given an integer array `nums`, you need to find one **continuous subarray** that if you only sort this subarray in ascending order, then the whole array will be sorted in ascending order.

Return *the shortest such subarray and output its length*.

Example 1:

```
Input: nums = [2,6,4,8,10,9,15]
Output: 5
Explanation: You need to sort [6, 4, 8, 10, 9] in ascending order to make the whole array sorted in ascending order.
```

Example 2:

```
Input: nums = [1,2,3,4]
Output: 0
```

Example 3:

```
Input: nums = [1]
Output: 0
```

Constraints:

- `1 <= nums.length <= 10<sup>4</sup>`
- `-10<sup>5</sup> <= nums[i] <= 10<sup>5</sup>`

Follow up: Can you solve it in `O(n)` time complexity?

Linear Time :

```
import sys
class Solution:
    def findUnsortedSubarray(self, nums: List[int]) -> int:
        leftMax = [-sys.maxsize]*len(nums)
        rightMin = [sys.maxsize]*len(nums)
        rightMin[-1] = nums[-1]
        leftMax[0] = nums[0]
        for i in range(1,len(nums)):
            leftMax[i] = max(leftMax[i-1],nums[i])
```

```
        for i in range(len(nums)-2,-1,-1):
            rightMin[i] = min(rightMin[i+1],nums[i])

        id1 = sys.maxsize
        id2 = -sys.maxsize
        ans = 0
        for i in range(len(nums)):
            if leftMax[i] != rightMin[i]:
                id1 = min(id1,i)
                id2 = max(id2,i)
                ans = max(ans,id2-id1+1)
        return ans
```

O(NlogN):

```
def findUnsortedSubarray(self, nums: List[int]) -> int:
        sorted_nums = nums.copy()
        sorted_nums = sorted(sorted_nums)
        start = len(nums)
        end = 0
        for i in range(0,len(nums)):
            if sorted_nums[i]!=nums[i]:
                start = min(i,start)
                end = max(end,i)
        if end-start<0:
            return 0
        else:
            return end-start +1
```