Problem # 581: Shortest Unsorted Continuous Subarray (Medium)

----------------------------------------------------------------------

<https://leetcode.com/problems/shortest-unsorted-continuous-subarray/>

My Solution:

<https://leetcode.com/problems/shortest-unsorted-continuous-subarray/discuss/1084483/Simple-Python-3-Solution-Runtime-beats-88.18>

Runtime beats 88.18%.

1. Let n be the length of nums.
2. Let sorted\_nums be nums sorted in ascending order.
3. If n is equal to 1 or if nums and sorted\_nums are equal then return 0, because there is nothing to sort.
4. Initialize subseq\_start to 1. This will hold the index of the beginning of the subarray.
5. Iterate through the array and compare every element from index 0 through n of nums with sorted\_nums. If they are not the same, subseq\_start will be equal to the index i, and then break out of the iteration.
6. Initialize subseq\_end to n-1, the last index of nums. This will hold the index of the end of the subarray.
7. Iterate from the end of the array to the subseq\_start. If they are not the same, subseq\_end will be equal to the index j, and then break out of the iteration.
8. Return the length of the subarray which is the difference of subseq\_end and subseq\_start and add 1 to it. Suppose we want index 2 through 5, we have 2, 3,4,5 and the number of elements = 4 which is 5 – 2 + 1.

class Solution:

def findUnsortedSubarray(self, nums: List[int]) -> int:

n = len(nums)

sorted\_nums = sorted(nums)

if n == 1 or nums == sorted(nums):

return 0

subseq\_start = 0

for i in range(n):

if nums[i] != sorted\_nums[i]:

subseq\_start = i

break

#print("subseq\_start = ", subseq\_start)

subseq\_end = n -1

for j in range(n-1, subseq\_start, -1):

if nums[j] != sorted\_nums[j]:

subseq\_end = j

break

#print("subseq\_end = ", subseq\_end)

return subseq\_end - subseq\_start + 1