Problem 3555: Smallest Subarray to Sort in Every Sliding Window

Problem Information

Difficulty: Medium
Acceptance Rate: 62.34%
Paid Only: Yes
Tags: Array, Two Pointers, Stack, Greedy, Sorting, Monotonic Stack

Problem Description

You are given an integer array `nums` and an integer `k`.

For each contiguous subarray of length `k`, determine the **minimum** length of a continuous segment that must be sorted so that the entire window becomes **non■decreasing** ; if the window is already sorted, its required length is zero.

Return an array of length `n – k + 1` where each element corresponds to the answer for its window.

Example 1:

Input: nums = [1,3,2,4,5], k = 3

Output: [2,2,0]

Explanation:

* `nums[0...2] = [1, 3, 2]`. Sort `[3, 2]` to get `[1, 2, 3]`, the answer is 2. * `nums[1...3] = [3, 2, 4]`. Sort `[3, 2]` to get `[2, 3, 4]`, the answer is 2. * `nums[2...4] = [2, 4, 5]` is already sorted, so the answer is 0.

Example 2:

Input: nums = [5,4,3,2,1], k = 4

Output: [4,4]

Explanation:

* `nums[0...3] = [5, 4, 3, 2]`. The whole subarray must be sorted, so the answer is 4. * `nums[1...4] = [4, 3, 2, 1]`. The whole subarray must be sorted, so the answer is 4.

Constraints:

* `1 <= nums.length <= 1000` * `1 <= k <= nums.length` * `1 <= nums[i] <= 106`

Code Snippets

C++:

```
class Solution {
public:
vector<int> minSubarraySort(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public int[] minSubarraySort(int[] nums, int k) {

}
}
```

Python3:

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