1063. Number of Valid Subarrays

Given an array `A` of integers, return the number of **non-empty continuous subarrays** that satisfy the following condition:

The leftmost element of the subarray is not larger than other elements in the subarray.

Example 1:

```
Input: [1,4,2,5,3]
Output: 11
Explanation: There are 11 valid subarrays: [1],[4],[2],[5],[3],[1,4],[2,5],
[1,4,2],[2,5,3],[1,4,2,5],[1,4,2,5,3].
```

Example 2:

```
Input: [3,2,1]
Output: 3
Explanation: The 3 valid subarrays are: [3],[2],[1].
```

Example 3:

```
Input: [2,2,2]
Output: 6
Explanation: There are 6 valid subarrays: [2],[2],[2],[2,2],[2,2],[2,2,2].
```

Note:

1. `1 <= A.length <= 50000`
2. `0 <= A[i] <= 100000`

Solution

What is actually asked in this problem: for every index we need to find the smallest , such that `nums[j] < nums[i]`. Let us traverse our numbers and put them into stack (in fact we will put indexes), such that we always have **non-decreasing** order in our stack. For example if we have `[1, 4, 2, 3, 5]`, then we have the following steps.

`[1]` `[1, 4]` `[1, 2]` `[1, 2, 5]` `[1, 2, 3]`

Note, that when we extract some element from stack, e.g we have `[1, 4]` and next element is `2`, it means that we found answer for number `4`, so we add it to final answer.

Complexity

Time and space complexity is O(N).

```
def validSubarray(arr):
    ans = 0
    n = len(arr)
    stack = []
    for i in range(len(arr) - 1, -1, -1):
        while len(stack) and arr[stack[-1]] >= arr[i]:
            stack.pop()

        ans = ans + (n - i if len(stack) == 0 else stack[-1] - i)
        stack.append(i)
    return ans


arr = [1, 3, 2]
print(validSubarray(arr))
```