1574. Shortest Subarray to be Removed to Make Array Sorted

Given an integer array `arr`, remove a subarray (can be empty) from `arr` such that the remaining elements in `arr` are **non-decreasing**.

Return *the length of the shortest subarray to remove*.

A **subarray** is a contiguous subsequence of the array.

Example 1:

```
Input: arr = [1,2,3,10,4,2,3,5]
Output: 3
Explanation: The shortest subarray we can remove is [10,4,2] of length 3. The
remaining elements after that will be [1,2,3,3,5] which are sorted.
Another correct solution is to remove the subarray [3,10,4].
```

Example 2:

```
Input: arr = [5,4,3,2,1]
Output: 4
Explanation: Since the array is strictly decreasing, we can only keep a single
element. Therefore we need to remove a subarray of length 4, either [5,4,3,2] or
[4,3,2,1].
```

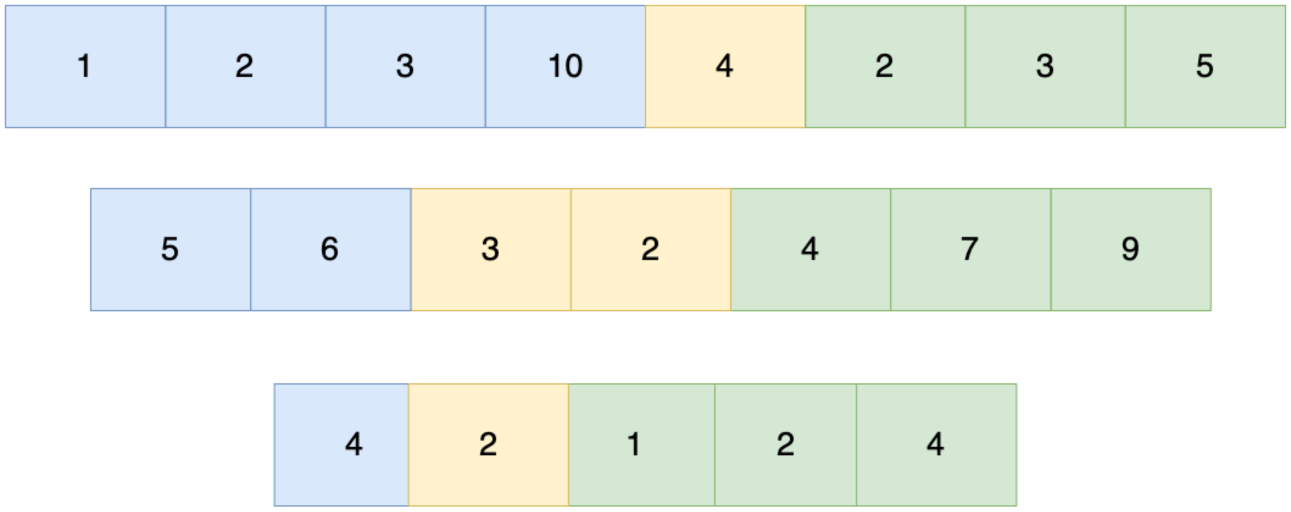
Example 3:

```
Input: arr = [1,2,3]
Output: 0
Explanation: The array is already non-decreasing. We do not need to remove any
elements.
```

Constraints:

- `1 <= arr.length <=` 10^5
- `0 <= arr[i] <=` 10^9

Intuition



We can think of arr as being composed of 3 parts. The first part is a block of numbers in sorted order (blue region in the image above), followed by a block of numbers that breaks the sorted order (yellow region), and then finally another block of numbers in sorted order (green region).

We have two edge cases to consider. The first is when the entire array is already sorted. In this case, no numbers need to be removed, so the subarray to be removed is empty. The second edge case is when the entire array is in reverse order (sorted in decreasing order), in which we remove everything except the first or last element.

For the nontrivial cases depicted above, we know that the subarray to remove resides somewhere in the middle of the array. Here, there can be multiple possibilities for what the middle elements can be. For the first example in the image, one option is to remove the block `[2, 3, 10, 4]`, leaving the remaining sorted sequence `[1, 2, 3, 5]`. Another option is to remove the block `[10, 4, 2]`, leaving another valid sequence `[1, 2, 3, 3, 5]`. The question then boils down to how we can find the smallest middle block of numbers to remove.

1574. Shortest Subarray to be Removed to Make Array Sorted

```
/*
   Two pointers
   Time complexity: O(n)
   Space complexity: O(1)
*/
class Solution {
   public:
      int findLengthOfShortestSubarray(vi& arr){
         int n=arr.size();

         // Find the left portion non-decreasing subarray
         // and the right portion non-decreasing subarray
         int end_left_portion=0,start_right_portion=n-1;
         while(end_left_portion<n-1 && arr[end_left_portion]<=arr[end_left_portion+1])
                  end_left_portion++;
         while(start_right_portion>0 && arr[start_right_portion]>=arr[start_right_portion-1])
                  start_right_portion--;

         // Array is already sorted
         if(end_left_portion>=start_right_portion) return 0;

         // find the minimum subarray to remove
         int ans=start_right_portion;
         int l=0,r=start_right_portion;
         while(l<=end_left_portion){
            while(r<n && arr[l]>arr[r]) r++;
            ans=std::min(ans,r-l-1);
            l++;
         }

         return ans;
      }
};
```