Shortest Unsorted Continuous Subarray

Given an integer array, 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, too.

You need to find the **shortest** such subarray and output its length.

Example 1:

```
Input: [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 a
rray sorted in ascending order.
```

Note:

1. Then length of the input array is in range [1, 10,000].
2. The input array may contain duplicates, so ascending order here means **.**

Solution 1

I use the variables `beg` and `end` to keep track of minimum subarray `A[beg...end]` which must be sorted for the entire array `A` to be sorted. If `end < beg < 0` at the end of the `for` loop, then the array is already fully sorted.

```
public int findUnsortedSubarray(int[] A) {
    int n = A.length, beg = -1, end = -2, min = A[n-1], max = A[0];
    for (int i=1;i<n;i++) {
      max = Math.max(max, A[i]);
      min = Math.min(min, A[n-1-i]);
      if (A[i] < max) end = i;
      if (A[n-1-i] > min) beg = n-1-i;
    }
    return end - beg + 1;
}
```

written by compton_scatter original link here

Solution 2

```
public class Solution {
    public int findUnsortedSubarray(int[] nums) {
        int n = nums.length;
        int[] temp = new int[n];
        for (int i = 0; i < n; i++) temp[i] = nums[i];
        Arrays.sort(temp);

        int start = 0;
        while (start < n  && nums[start] == temp[start]) start++;

        int end = n - 1;
        while (end > start  && nums[end] == temp[end]) end--;

        return end - start + 1;
    }
}
```

written by shawngao original link here

Solution 3

```
int findUnsortedSubarray(vector<int>& nums) {
     int shortest = 0;

     int left = 0, right = nums.size() - 1;
     while (left < nums.size() - 1 && nums[left] <= nums[left + 1]) { left++; }
     while (right > 0 && nums[right] >= nums[right - 1]) { right--; };

     if (right > left) {
         int vmin = INT_MAX, vmax = INT_MIN;
         for (int i = left; i <= right; ++i) {
             if (nums[i] > vmax) {
                 vmax = nums[i];
             }
             if (nums[i] < vmin) {
                 vmin = nums[i];
             }
         }

         while (left >= 0 && nums[left] > vmin) { left--; };
         while (right < nums.size() && nums[right] < vmax) { right++; };

         shortest = right - left - 1;
     }

     return shortest;
 }
```

written by Xnming original link here