1752. Check if Array Is Sorted and Rotated

Given an array `nums`, return `true` *if the array was originally sorted in non-decreasing order, then rotated* ***some*** *number of positions (including zero).* Otherwise, return `false`.

There may be **duplicates** in the original array.

Note: An array `A` rotated by `x` positions results in an array `B` of the same length such that `A[i] == B[(i+x) % A.length]`, where `%` is the modulo operation.

Example 1:

```
Input: nums = [3,4,5,1,2]
Output: true
Explanation: [1,2,3,4,5] is the original sorted array.
You can rotate the array by x = 3 positions to begin on the the element of value 3: [3,4,5,1,2].
```

Example 2:

```
Input: nums = [2,1,3,4]
Output: false
Explanation: There is no sorted array once rotated that can make nums.
```

Example 3:

```
Input: nums = [1,2,3]
Output: true
Explanation: [1,2,3] is the original sorted array.
You can rotate the array by x = 0 positions (i.e. no rotation) to make nums.
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`

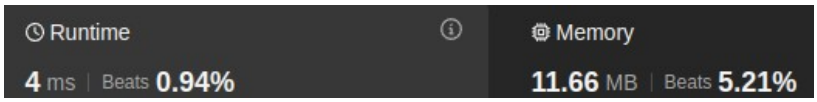
Overview

We need to find whether the given integer array `nums` could represent a sorted array that has been rotated some number of times. A sorted array is defined as one arranged in non-decreasing order, meaning each element is less than or equal to the next. A rotation involves shifting a contiguous block of elements to the back of the array, preserving the relative order of all elements.

For example, `[3, 4, 5, 1, 2]` is a rotated version of the sorted array `[1, 2, 3, 4, 5]`. On the other hand, `[3, 4, 2, 1, 5]` is not a valid rotation of any sorted array because the order of elements is not preserved.

1752. Check if Array Is Sorted and Rotated

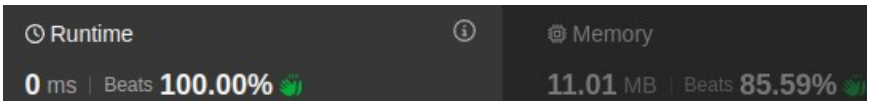
```
/*
   Compare with sorted array
   Time complexity: O(nlogn+n^2)
   Space complexity: O(n+logn)
*/
```




```
class Solution {
   public:
      bool check(std::vector<int>& nums) {
         int n=nums.size();
         std::vector<int> sorted=nums;
         std::sort(sorted.begin(),sorted.end());
         for(int rot_index=0;rot_index<n;++rot_index){
            int i=0;
            while(i<n && nums[(rot_index+i)%n]==sorted[i]) i++;
            if(i==n) return true;
         }
         return false;
      }
};
```

1752. Check if Array Is Sorted and Rotated

```
/*
   Counting bad pairs
   Time complexity: O(n)
   Space complexity: O(1)
*/
```




```
class Solution {
   public:
      bool check(std::vector<int>& nums) {
         int n=nums.size();
         int count=0;
         for(int i=0;i<n-1;++i){
            if(nums[i+1]<nums[i]) count++;
         }
         count+=nums[0]<nums[n-1];
         return count<2;
      }
};
```