Check If Array Sorted & Rotated LeetCode

Given an array of integers, determine whether the array is sorted and rotated. An array is considered sorted and rotated if it is sorted in non-decreasing order and has been rotated at some pivot point.

Explanation of the Code: The provided code includes two approaches (**checkSortedRotated** and **isSortedRotated**) to solve the problem of checking if an array is sorted and rotated.

Input Array: {4, 5, 6, 7, 1, 2, 3} Explanation: The array is sorted in non-decreasing order and has been rotated at index 3 (pivot point). Therefore, it is a sorted and rotated array.

Approach 1: Check if the array is sorted and rotated

- This approach uses a loop to iterate through the array and counts the number of decreasing pairs.
- It starts from the second element and compares it with the previous element. If the previous element is greater, it increments the **count** variable.
- After iterating through the entire array, it checks if the last element is greater than the first element. If so, it increments the **count** variable.
- Finally, it returns **true** if the count is less than or equal to 1, indicating that there is at most one decreasing pair in the array.

The approach has a time complexity of O(n), where n is the size of the input array. This is because it iterates through the entire array once.

The space complexity is O(1) since they only use a constant amount of extra space for variables.

Approach 2: Check if the array is sorted and rotated within a single loop & modulus operator.

- This approach also uses a loop to iterate through the array, but it employs the modulus operator to handle the circular nature of rotation.
- It compares the current element with the next element (using **(i+1) % size** to wrap around to the beginning of the array).
- If a decreasing pair is found, it increments the **count** variable.
- Additionally, it checks if the count exceeds 1 during the iteration. If so, it immediately returns **false** to indicate that the array is not sorted and rotated.
- If the loop completes without finding more than one decreasing pair, it returns **true**.

The approach has a time complexity of O(n), where n is the size of the input array. This is because it iterates through the entire array once.

The space complexity is O(1) since they only use a constant amount of extra space for variables.

Which Approach to Use:

- Both approaches are valid and provide the same result. You can choose either approach based on your preference or coding style.
- Approach 2 (**isSortedRotated**) is slightly more optimized as it eliminates the need for an extra check at the end of the loop.
- If you want to check if an array is sorted and rotated, you can use either approach depending on your requirements and coding preferences.