189. Rotate Array

Medium 7487 1157 Add to List Share

Given an array, rotate the array to the right by `k` steps, where `k` is non-negative.

Example 1:

```
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
```

Example 2:

```
Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]
```

Constraints:

- `1 <= nums.length <= 10^5`
- `-2^31 <= nums[i] <= 2^31 - 1`
- `0 <= k <= 10^5`

Follow up:

- Try to come up with as many solutions as you can. There are at least **three** different ways to solve this problem.
- Could you do it in-place with `O(1)` extra space?

Java Autocomplete

```java
//Constant space solution using rotation
class Solution {

    private void reverseArray(int[] array, int startIndex, int endIndex){
        while(startIndex <= endIndex){
            int temp = array[startIndex];

            array[startIndex++] = array[endIndex];
            array[endIndex--] = temp;
        }
    }

    public void rotate(int[] nums, int k) {

        k = k % nums.length;

        reverseArray(nums, 0, nums.length - 1);

        reverseArray(nums, 0, k-1);
        reverseArray(nums, k, nums.length - 1);
    }
}
```

Your previous code was restored from your local storage. Reset to default

977. Squares of a Sorted Array

Easy 4402 143 Add to List Share

Given an integer array `nums` sorted in **non-decreasing** order, return *an array of **the squares of each number** sorted in non-decreasing order.*

Example 1:

```
Input: nums = [-4,-1,0,3,10]
Output: [0,1,9,16,100]
Explanation: After squaring, the array becomes [16,1,0,9,100].
After sorting, it becomes [0,1,9,16,100].
```

Example 2:

```
Input: nums = [-7,-3,2,3,11]
Output: [4,9,9,49,121]
```

Constraints:

- `1 <= nums.length <= 10^4`
- `-10^4 <= nums[i] <= 10^4`
- `nums` is sorted in **non-decreasing** order.

Follow up: Squaring each element and sorting the new array is very trivial, could you find an `O(n)` solution using a different approach?

Accepted 812,703 Submissions 1,137,011

Seen this question in a real interview before? Yes No

Companies

Related Topics


```java
class Solution {
    public int[] sortedSquares(int[] A) {

        int n = A.length;
        int[] result = new int[n];
        int i = 0, j = n - 1;

        for (int p = n - 1; p >= 0; p--) {
            if (Math.abs(A[i]) > Math.abs(A[j])) {
                result[p] = A[i] * A[i];
                i++;
            } else {
                result[p] = A[j] * A[j];
                j--;
            }
        }

        return result;
    }
}
```

Your previous code was restored from your local storage. Reset to default