Problem 189: Rotate Array

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given an integer array

nums

, 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?

Code Snippets

C++:

```
class Solution {
public:
void rotate(vector<int>& nums, int k) {

}
};
```

Java:

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

}
}
```

Python3:

```
class Solution:
def rotate(self, nums: List[int], k: int) -> None:
"""
Do not return anything, modify nums in-place instead.
"""
```

Python:

```
class Solution(object):
def rotate(self, nums, k):
"""
:type nums: List[int]
:type k: int
:rtype: None Do not return anything, modify nums in-place instead.
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @param {number} k
* @return {void} Do not return anything, modify nums in-place instead.
*/
var rotate = function(nums, k) {

};
```

TypeScript:

```
/**
Do not return anything, modify nums in-place instead.
*/
function rotate(nums: number[], k: number): void {

};
```

C#:

```
public class Solution {
public void Rotate(int[] nums, int k) {

}
}
```

C:

```
void rotate(int* nums, int numsSize, int k) {

}
```

Go:

```
func rotate(nums []int, k int) {

}
```

Kotlin:

```
class Solution {
fun rotate(nums: IntArray, k: Int): Unit {

}
}
```

Swift:

```
class Solution {
func rotate(_ nums: inout [Int], _ k: Int) {

}
}
```

Rust:

```
impl Solution {
pub fn rotate(nums: &mut Vec<i32>, k: i32) {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Void} Do not return anything, modify nums in-place instead.
def rotate(nums, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return NULL
*/
function rotate(&$nums, $k) {

}
}
```

Dart:

```
class Solution {
void rotate(List<int> nums, int k) {

}
}
```

Scala:

```
object Solution {
def rotate(nums: Array[Int], k: Int): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
```

```
void rotate(vector<int>& nums, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Rotate Array
Difficulty: Medium
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def rotate(self, nums: List[int], k: int) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def rotate(self, nums, k):
"""
:type nums: List[int]
:type k: int
:rtype: None Do not return anything, modify nums in-place instead.
"""
```

JavaScript Solution:

```
/**
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @param {number} k
* @return {void} Do not return anything, modify nums in-place instead.
*/
var rotate = function(nums, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
Do not return anything, modify nums in-place instead.
*/
function rotate(nums: number[], k: number): void {

};
```

C# Solution:

```
/*
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public void Rotate(int[] nums, int k) {

}
}
```

C Solution:

```
/*
* Problem: Rotate Array
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

void rotate(int* nums, int numsSize, int k) {

}
```

Go Solution:

```
// Problem: Rotate Array
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func rotate(nums []int, k int) {

}
```

Kotlin Solution:

```
class Solution {
fun rotate(nums: IntArray, k: Int): Unit {

}
}
```

Swift Solution:

```
class Solution {
func rotate(_ nums: inout [Int], _ k: Int) {

}
}
```

Rust Solution:

```
// Problem: Rotate Array
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn rotate(nums: &mut Vec<i32>, k: i32) {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} k
# @return {Void} Do not return anything, modify nums in-place instead.
def rotate(nums, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $k
* @return NULL
*/
function rotate(&$nums, $k) {

}
}
```

Dart Solution:

```
class Solution {
void rotate(List<int> nums, int k) {

}
}
```

Scala Solution:

```
object Solution {
def rotate(nums: Array[Int], k: Int): Unit = {

}
}
```