Problem 283: Move Zeroes

Problem Information

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

Problem Description

Given an integer array

nums

, move all

0

's to the end of it while maintaining the relative order of the non-zero elements.

Note

that you must do this in-place without making a copy of the array.

Example 1:

Input:

nums = [0,1,0,3,12]

Output:

[1,3,12,0,0]

Example 2:

Input:

nums = [0]

Output:

[0]

Constraints:

1 <= nums.length <= 10

4

-2

31

<= nums[i] <= 2

31

- 1

Follow up:

Could you minimize the total number of operations done?

Code Snippets

C++:

```
class Solution {
public:
void moveZeroes(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public void moveZeroes(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public void MoveZeroes(int[] nums) {

}
}
```

C:

```
void moveZeroes(int* nums, int numsSize) {

}
```

Go:

```
func moveZeroes(nums []int) {

}
```

Kotlin:

```
class Solution {
fun moveZeroes(nums: IntArray): Unit {

}
}
```

Swift:

```
class Solution {
func moveZeroes(_ nums: inout [Int]) {

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
void moveZeroes(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def moveZeroes(nums: Array[Int]): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
* 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 moveZeroes(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
* 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 moveZeroes(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Move Zeroes
Difficulty: Easy
Tags: array
```

```
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 moveZeroes(self, nums: List[int]) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
* 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
* @return {void} Do not return anything, modify nums in-place instead.
*/
var moveZeroes = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
* 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 moveZeroes(nums: number[]): void {

};
```

C# Solution:

```
/*
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
* 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 MoveZeroes(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Move Zeroes
* Difficulty: Easy
* Tags: array
*
```

```
* 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 moveZeroes(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Move Zeroes
// Difficulty: Easy
// Tags: array
//
// 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 moveZeroes(nums []int) {

}
```

Kotlin Solution:

```
class Solution {
fun moveZeroes(nums: IntArray): Unit {

}
}
```

Swift Solution:

```
class Solution {
func moveZeroes(_ nums: inout [Int]) {

}
}
```

Rust Solution:

```
// Problem: Move Zeroes
// Difficulty: Easy
// Tags: array
//
// 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 move_zeroes(nums: &mut Vec<i32>) {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
void moveZeroes(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def moveZeroes(nums: Array[Int]): Unit = {

}
}
```