Problem 1089: Duplicate Zeros

Problem Information

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

Problem Description

Given a fixed-length integer array

arr

, duplicate each occurrence of zero, shifting the remaining elements to the right.

Note

that elements beyond the length of the original array are not written. Do the above modifications to the input array in place and do not return anything.

Example 1:

Input:

arr = [1,0,2,3,0,4,5,0]

Output:

[1,0,0,2,3,0,0,4]

Explanation:

After calling your function, the input array is modified to: [1,0,0,2,3,0,0,4]

Example 2:

Input:

arr = [1,2,3]

Output:

[1,2,3]

Explanation:

After calling your function, the input array is modified to: [1,2,3]

Constraints:

1 <= arr.length <= 10

4

0 <= arr[i] <= 9

Code Snippets

C++:

```
class Solution {
public:
void duplicateZeros(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public void duplicateZeros(int[] arr) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public void DuplicateZeros(int[] arr) {

}
}
```

C:

```
void duplicateZeros(int* arr, int arrSize) {

}
```

Go:

```
func duplicateZeros(arr []int) {

}
```

Kotlin:

```
class Solution {
fun duplicateZeros(arr: IntArray): Unit {

}
}
```

Swift:

```
class Solution {
func duplicateZeros(_ arr: inout [Int]) {

}
}
```

Rust:

```
impl Solution {
pub fn duplicate_zeros(arr: &mut Vec<i32>) {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return NULL
*/
function duplicateZeros(&$arr) {

}
}
```

Dart:

```
class Solution {
void duplicateZeros(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def duplicateZeros(arr: Array[Int]): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Duplicate Zeros
* 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 duplicateZeros(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Duplicate Zeros
* 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 duplicateZeros(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Duplicate Zeros
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 duplicateZeros(self, arr: List[int]) -> None:
```

```
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Duplicate Zeros
* 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[]} arr
* @return {void} Do not return anything, modify arr in-place instead.
*/
var duplicateZeros = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Duplicate Zeros
* 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 arr in-place instead.
*/
function duplicateZeros(arr: number[]): void {

};
```

C# Solution:

```
/*
* Problem: Duplicate Zeros
* 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 DuplicateZeros(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Duplicate Zeros
* 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 duplicateZeros(int* arr, int arrSize) {
```

```
}
```

Go Solution:

```
// Problem: Duplicate Zeros
// 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 duplicateZeros(arr []int) {

}
```

Kotlin Solution:

```
class Solution {
fun duplicateZeros(arr: IntArray): Unit {

}
}
```

Swift Solution:

```
class Solution {
func duplicateZeros(_ arr: inout [Int]) {

}
}
```

Rust Solution:

```
// Problem: Duplicate Zeros
// 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 duplicate_zeros(arr: &mut Vec<i32>) {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return NULL
*/
function duplicateZeros(&$arr) {

}
}
```

Dart Solution:

```
class Solution {
void duplicateZeros(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def duplicateZeros(arr: Array[Int]): Unit = {
```

```
}
}
```