Problem 2134: Minimum Swaps to Group All 1's Together II

Problem Information

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

Problem Description

A

swap

is defined as taking two

distinct

positions in an array and swapping the values in them.

A

circular

array is defined as an array where we consider the

first

element and the

last

element to be

adjacent

.

Given a

binary

circular

array

nums

, return

the minimum number of swaps required to group all

1

's present in the array together at

any location

.

Example 1:

Input:

nums = [0,1,0,1,1,0,0]

Output:

1

Explanation:

Here are a few of the ways to group all the 1's together: [0, 0

,

1

,1,1,0,0] using 1 swap. [0,1,

1

,1,

0

,0,0] using 1 swap. [1,1,0,0,0,0,1] using 2 swaps (using the circular property of the array). There is no way to group all 1's together with 0 swaps. Thus, the minimum number of swaps required is 1.

Example 2:

Input:

nums = [0,1,1,1,0,0,1,1,0]

Output:

2

Explanation:

Here are a few of the ways to group all the 1's together: [1,1,1,0,0,0,0,1,1] using 2 swaps (using the circular property of the array). [1,1,1,1,1,0,0,0,0] using 2 swaps. There is no way to group all 1's together with 0 or 1 swaps. Thus, the minimum number of swaps required is 2.

Example 3:

Input:

nums = [1,1,0,0,1]

Output:

0

Explanation:

All the 1's are already grouped together due to the circular property of the array. Thus, the minimum number of swaps required is 0.

Constraints:

1 <= nums.length <= 10

5

nums[i]

is either

0

or

1

.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int minSwaps(int[] nums) {

}
}
```

Python3:

```
class Solution:
def minSwaps(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def minSwaps(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var minSwaps = function(nums) {

};
```

TypeScript:

```
function minSwaps(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int MinSwaps(int[] nums) {
}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun minSwaps(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minSwaps(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_swaps(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def min_swaps(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function minSwaps($nums) {

}
}
```

Dart:

```
class Solution {
int minSwaps(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

```
defmodule Solution do
@spec min_swaps(nums :: [integer]) :: integer
def min_swaps(nums) do

end
end
```

Erlang:

```
-spec min_swaps(Nums :: [integer()]) -> integer().
min_swaps(Nums) ->
.
```

Racket:

```
(define/contract (min-swaps nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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:
int minSwaps(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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 int minSwaps(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Swaps to Group All 1's Together II
Difficulty: Medium
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 minSwaps(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minSwaps(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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 {number}
*/
var minSwaps = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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
*/

function minSwaps(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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 int MinSwaps(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together II
* Difficulty: Medium
* 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
*/

int minSwaps(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Swaps to Group All 1's Together II
// Difficulty: Medium
// 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 minSwaps(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minSwaps(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minSwaps(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Swaps to Group All 1's Together II
// Difficulty: Medium
// 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 min_swaps(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def min_swaps(nums)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $nums
* @return Integer
*/
function minSwaps($nums) {

}
}
```

Dart Solution:

```
class Solution {
int minSwaps(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_swaps(nums :: [integer]) :: integer
def min_swaps(nums) do

end
end
```

Erlang Solution:

```
-spec min_swaps(Nums :: [integer()]) -> integer().
min_swaps(Nums) ->
.
```

Racket Solution:

```
(define/contract (min-swaps nums)
(-> (listof exact-integer?) exact-integer?)
)
```