Problem 1151: Minimum Swaps to Group All 1's Together

Problem Information

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

Problem Description

Given a binary array

data

, return the minimum number of swaps required to group all

1

’s present in the array together in

any place

in the array.

Example 1:

Input:

data = [1,0,1,0,1]

Output:

1

Explanation:

There are 3 ways to group all 1's together: [1,1,1,0,0] using 1 swap. [0,1,1,1,0] using 2 swaps. [0,0,1,1,1] using 1 swap. The minimum is 1.

Example 2:

Input:

data = [0,0,0,1,0]

Output:

0

Explanation:

Since there is only one 1 in the array, no swaps are needed.

Example 3:

Input:

data = [1,0,1,0,1,0,0,1,1,0,1]

Output:

3

Explanation:

One possible solution that uses 3 swaps is [0,0,0,0,0,1,1,1,1,1,1].

Constraints:

1 <= data.length <= 10

5

data[i]

is either

0

or

1

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int MinSwaps(int[] data) {

}
}
```

C:

```
int minSwaps(int* data, int dataSize) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together
* 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>& data) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together
* 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[] data) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Swaps to Group All 1's Together
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, data: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together
* 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[]} data
* @return {number}
*/
var minSwaps = function(data) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Swaps to Group All 1's Together
* 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(data: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together
* 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[] data) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Swaps to Group All 1's Together
* 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* data, int dataSize) {

}
```

Go Solution:

```
// Problem: Minimum Swaps to Group All 1's Together
// 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(data []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Swaps to Group All 1's Together
// 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(data: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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