Problem 1864: Minimum Number of Swaps to Make the Binary String Alternating

Problem Information

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

Problem Description

Given a binary string

s

, return

the

minimum

number of character swaps to make it

alternating

, or

-1

if it is impossible.

The string is called

alternating

if no two adjacent characters are equal. For example, the strings

"010"

and

"1010"

are alternating, while the string

"0100"

is not.

Any two characters may be swapped, even if they are

not adjacent

.

Example 1:

Input:

s = "111000"

Output:

1

Explanation:

Swap positions 1 and 4: "1

1

10

0

0" -> "1

0

10

1

0" The string is now alternating.

Example 2:

Input:

s = "010"

Output:

0

Explanation:

The string is already alternating, no swaps are needed.

Example 3:

Input:

s = "1110"

Output:

-1

Constraints:

1 <= s.length <= 1000

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
int minSwaps(string s) {

}
};
```

Java:

```
class Solution {
public int minSwaps(String s) {

}
}
```

Python3:

```
class Solution:
def minSwaps(self, s: str) -> int:
```

Python:

```
class Solution(object):
def minSwaps(self, s):
"""
:type s: str
```

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

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var minSwaps = function(s) {

};
```

TypeScript:

```
function minSwaps(s: string): number {

};
```

C#:

```
public class Solution {
public int MinSwaps(string s) {

}
}
```

C:

```
int minSwaps(char* s) {

}
```

Go:

```
func minSwaps(s string) int {

}
```

Kotlin:

```
class Solution {
fun minSwaps(s: String): Int {

}
}
```

Swift:

```
class Solution {
func minSwaps(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_swaps(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def min_swaps(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function minSwaps($s) {

}
}
```

Dart:

```
class Solution {
int minSwaps(String s) {

}
}
```

Scala:

```
object Solution {
def minSwaps(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_swaps(s :: String.t) :: integer
def min_swaps(s) do

end
end
```

Erlang:

```
-spec min_swaps(S :: unicode:unicode_binary()) -> integer().
min_swaps(S) ->
.
```

Racket:

```
(define/contract (min-swaps s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int minSwaps(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int minSwaps(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Swaps to Make the Binary String Alternating
Difficulty: Medium
Tags: string, greedy
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def minSwaps(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minSwaps(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @return {number}
*/
var minSwaps = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function minSwaps(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* 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(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Swaps to Make the Binary String Alternating
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

int minSwaps(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Number of Swaps to Make the Binary String Alternating
// Difficulty: Medium
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minSwaps(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minSwaps(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minSwaps(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Swaps to Make the Binary String Alternating
// Difficulty: Medium
// Tags: string, greedy
```

```
//
// Approach: String manipulation with hash map or two pointers
// 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(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def min_swaps(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function minSwaps($s) {

}
}
```

Dart Solution:

```
class Solution {
int minSwaps(String s) {

}
}
```

Scala Solution:

```
object Solution {
def minSwaps(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_swaps(s :: String.t) :: integer
def min_swaps(s) do

end
end
```

Erlang Solution:

```
-spec min_swaps(S :: unicode:unicode_binary()) -> integer().
min_swaps(S) ->
.
```

Racket Solution:

```
(define/contract (min-swaps s)
(-> string? exact-integer?)
)
```