Problem 1963: Minimum Number of Swaps to Make the String Balanced

Problem Information

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

Problem Description

You are given a

0-indexed

string

s

of

even

length

n

. The string consists of

exactly

n / 2

opening brackets

'['

and

n / 2

closing brackets

']'

.

A string is called

balanced

if and only if:

It is the empty string, or

It can be written as

AB

, where both

A

and

B

are

balanced

strings, or

It can be written as

[C]

, where

C

is a

balanced

string.

You may swap the brackets at

any

two indices

any

number of times.

Return

the

minimum

number of swaps to make

s

balanced

.

Example 1:

Input:

s = "][]["

Output:

1

Explanation:

You can make the string balanced by swapping index 0 with index 3. The resulting string is "[[]]".

Example 2:

Input:

s = "]]][[["

Output:

2

Explanation:

You can do the following to make the string balanced: - Swap index 0 with index 4. s = "[]][[". - Swap index 1 with index 5. s = "[[][]]". The resulting string is "[[][]]".

Example 3:

Input:

s = "[]"

Output:

0

Explanation:

The string is already balanced.

Constraints:

n == s.length

2 <= n <= 10

6

n

is even.

s[i]

is either

'['

or

']'

.

The number of opening brackets

'['

equals

n / 2

, and the number of closing brackets

']'

equals

n / 2

.

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 String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Swaps to Make the String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Swaps to Make the String Balanced
Difficulty: Medium
Tags: array, string, greedy, stack

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, 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 String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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 {string} s
* @return {number}
*/
var minSwaps = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Swaps to Make the String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Swaps to Make the String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Swaps to Make the String Balanced
* Difficulty: Medium
* Tags: array, string, greedy, stack
*
* 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(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Number of Swaps to Make the String Balanced
// Difficulty: Medium
// Tags: array, string, greedy, stack
//
// 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(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 String Balanced
// Difficulty: Medium
// Tags: array, string, greedy, stack
//
// 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(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?)
)
```