Problem 1869: Longer Contiguous Segments of Ones than Zeros

Problem Information

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

Problem Description

Given a binary string

s

, return

true

if the

longest

contiguous segment of

1

'

s is

strictly longer

than the

longest

contiguous segment of

0

'

s in

s

, or return

false

otherwise

.

For example, in

s = "

11

01

000

10"

the longest continuous segment of

1

s has length

2

, and the longest continuous segment of

0

s has length

3

.

Note that if there are no

0

's, then the longest continuous segment of

0

's is considered to have a length

0

. The same applies if there is no

1

's.

Example 1:

Input:

s = "1101"

Output:

true

Explanation:

The longest contiguous segment of 1s has length 2: "

11

01" The longest contiguous segment of 0s has length 1: "11

0

1" The segment of 1s is longer, so return true.

Example 2:

Input:

s = "111000"

Output:

false

Explanation:

The longest contiguous segment of 1s has length 3: "

111

000" The longest contiguous segment of 0s has length 3: "111

000

" The segment of 1s is not longer, so return false.

Example 3:

Input:

s = "110100010"

Output:

false

Explanation:

The longest contiguous segment of 1s has length 2: "

11

0100010" The longest contiguous segment of 0s has length 3: "1101

000

10" The segment of 1s is not longer, so return false.

Constraints:

1 <= s.length <= 100

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

```
class Solution {
public:
```

```
bool checkZeroOnes(string s) {

}
};
```

Java:

```
class Solution {
public boolean checkZeroOnes(String s) {

}
}
```

Python3:

```
class Solution:
def checkZeroOnes(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def checkZeroOnes(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var checkZeroOnes = function(s) {

};
```

TypeScript:

```
function checkZeroOnes(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool CheckZeroOnes(string s) {

}
}
```

C:

```
bool checkZeroOnes(char* s) {

}
```

Go:

```
func checkZeroOnes(s string) bool {

}
```

Kotlin:

```
class Solution {
fun checkZeroOnes(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func checkZeroOnes(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_zero_ones(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def check_zero_ones(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function checkZeroOnes($s) {

}
}
```

Dart:

```
class Solution {
bool checkZeroOnes(String s) {

}
}
```

Scala:

```
object Solution {
def checkZeroOnes(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_zero_ones(s :: String.t) :: boolean
def check_zero_ones(s) do
```

```
end
end
```

Erlang:

```
-spec check_zero_ones(S :: unicode:unicode_binary()) -> boolean().
check_zero_ones(S) ->
.
```

Racket:

```
(define/contract (check-zero-ones s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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:
bool checkZeroOnes(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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 boolean checkZeroOnes(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Longer Contiguous Segments of Ones than Zeros
Difficulty: Easy
Tags: string

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 checkZeroOnes(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkZeroOnes(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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 {boolean}
*/
var checkZeroOnes = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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 checkZeroOnes(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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 bool CheckZeroOnes(string s) {

}
}
```

C Solution:

```
/*
* Problem: Longer Contiguous Segments of Ones than Zeros
* Difficulty: Easy
* Tags: string
*
* 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
*/

bool checkZeroOnes(char* s) {

}
```

Go Solution:

```
// Problem: Longer Contiguous Segments of Ones than Zeros
// Difficulty: Easy
// Tags: string
//
// 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 checkZeroOnes(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkZeroOnes(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkZeroOnes(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Longer Contiguous Segments of Ones than Zeros
// Difficulty: Easy
// Tags: string
//
// 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 check_zero_ones(s: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def check_zero_ones(s)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $s
* @return Boolean
*/
function checkZeroOnes($s) {

}
}
```

Dart Solution:

```
class Solution {
bool checkZeroOnes(String s) {

}
}
```

Scala Solution:

```
object Solution {
def checkZeroOnes(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_zero_ones(s :: String.t) :: boolean
def check_zero_ones(s) do

end
end
```

Erlang Solution:

```
-spec check_zero_ones(S :: unicode:unicode_binary()) -> boolean().
check_zero_ones(S) ->
.
```

Racket Solution:

```
(define/contract (check-zero-ones s)
(-> string? boolean?)
)
```