Problem 2380: Time Needed to Rearrange a Binary String

Problem Information

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

Problem Description

You are given a binary string

s

. In one second,

all

occurrences of

"01"

are

simultaneously

replaced with

"10"

. This process

repeats

until no occurrences of

"01"

exist.

Return

the number of seconds needed to complete this process.

Example 1:

Input:

s = "0110101"

Output:

4

Explanation:

After one second, s becomes "1011010". After another second, s becomes "1101100". After the third second, s becomes "1110100". After the fourth second, s becomes "1111000". No occurrence of "01" exists any longer, and the process needed 4 seconds to complete, so we return 4.

Example 2:

Input:

s = "11100"

Output:

0

Explanation:

No occurrence of "01" exists in s, and the processes needed 0 seconds to complete, so we return 0.

Constraints:

1 <= s.length <= 1000

s[i]

is either

'0'

or

'1'

.

Follow up:

Can you solve this problem in O(n) time complexity?

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int secondsToRemoveOccurrences(char* s) {

}
```

Go:

```
func secondsToRemoveOccurrences(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (seconds-to-remove-occurrences s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int secondsToRemoveOccurrences(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Time Needed to Rearrange a Binary String
Difficulty: Medium
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function secondsToRemoveOccurrences(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int SecondsToRemoveOccurrences(string s) {

}
}
```

C Solution:

```
/*
* Problem: Time Needed to Rearrange a Binary String
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int secondsToRemoveOccurrences(char* s) {

}
```

Go Solution:

```
// Problem: Time Needed to Rearrange a Binary String
// Difficulty: Medium
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func secondsToRemoveOccurrences(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Time Needed to Rearrange a Binary String
// Difficulty: Medium
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (seconds-to-remove-occurrences s)
(-> string? exact-integer?)
)
```