Problem 3499: Maximize Active Section with Trade I

Problem Information

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

Problem Description

You are given a binary string

s

of length

n

, where:

'1'

represents an

active

section.

'0'

represents an

inactive

section.

You can perform

at most one trade

to maximize the number of active sections in

s

. In a trade, you:

Convert a contiguous block of

'1'

s that is surrounded by

'0'

s to all

'0'

s.

Afterward, convert a contiguous block of

'0'

s that is surrounded by

'1'

s to all

'1'

s.

Return the

maximum

number of active sections in

s

after making the optimal trade.

Note:

Treat

s

as if it is

augmented

with a

'1'

at both ends, forming

t = '1' + s + '1'

. The augmented

'1'

s

do not

contribute to the final count.

Example 1:

Input:

s = "01"

Output:

1

Explanation:

Because there is no block of

'1'

s surrounded by

'0'

s, no valid trade is possible. The maximum number of active sections is 1.

Example 2:

Input:

s = "0100"

Output:

4

Explanation:

String

"0100"

→ Augmented to

"101001"

.

Choose

"0100"

, convert

"10

1

001"

→

"1

0000

1"

→

"1

1111

1"

.

The final string without augmentation is

"1111"

. The maximum number of active sections is 4.

Example 3:

Input:

s = "1000100"

Output:

7

Explanation:

String

"1000100"

→ Augmented to

"110001001"

.

Choose

"000100"

, convert

"11000

1

001"

→

"11

000000

1"

→

"11

111111

1"

.

The final string without augmentation is

"1111111"

. The maximum number of active sections is 7.

Example 4:

Input:

s = "01010"

Output:

4

Explanation:

String

"01010"

→ Augmented to

"1010101"

.

Choose

"010"

, convert

"10

1

0101"

→

"1

000

101"

→

"1

111

101"

.

The final string without augmentation is

"11110"

. The maximum number of active sections is 4.

Constraints:

1 <= n == s.length <= 10

5

s[i]

is either

'0'

or

'1'

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int maxActiveSectionsAfterTrade(char* s) {

}
```

Go:

```
func maxActiveSectionsAfterTrade(s string) int {
```

```
}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
```

```
function maxActiveSectionsAfterTrade($s) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-active-sections-after-trade s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
* 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:
int maxActiveSectionsAfterTrade(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
* 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 int maxActiveSectionsAfterTrade(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximize Active Section with Trade I
Difficulty: Medium
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 maxActiveSectionsAfterTrade(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
* 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 {number}
*/
var maxActiveSectionsAfterTrade = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
* 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 maxActiveSectionsAfterTrade(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
* 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 int MaxActiveSectionsAfterTrade(string s) {

}
}
```

C Solution:

```
/*
* Problem: Maximize Active Section with Trade I
* Difficulty: Medium
```

```
* 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
*/

int maxActiveSectionsAfterTrade(char* s) {

}
```

Go Solution:

```
// Problem: Maximize Active Section with Trade I
// Difficulty: Medium
// 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 maxActiveSectionsAfterTrade(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximize Active Section with Trade I
// Difficulty: Medium
// 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 max_active_sections_after_trade(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-active-sections-after-trade s)
(-> string? exact-integer?)
)
```