Problem 1422: Maximum Score After Splitting a String

Problem Information

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

Problem Description

Given a string

s

of zeros and ones,

return the maximum score after splitting the string into two

non-empty

substrings

(i.e.

left

substring and

right

substring).

The score after splitting a string is the number of

zeros

in the

left

substring plus the number of

ones

in the

right

substring.

Example 1:

Input:

s = "011101"

Output:

5

Explanation:

All possible ways of splitting s into two non-empty substrings are: left = "0" and right = "11101", score = 1 + 4 = 5 left = "01" and right = "1101", score = 1 + 3 = 4 left = "011" and right = "101", score = 1 + 2 = 3 left = "0111" and right = "01", score = 1 + 1 = 2 left = "01110" and right = "1", score = 2 + 1 = 3

Example 2:

Input:

s = "00111"

Output:

5

Explanation:

When left = "00" and right = "111", we get the maximum score = 2 + 3 = 5

Example 3:

Input:

s = "1111"

Output:

3

Constraints:

2 <= s.length <= 500

The string

s

consists of characters

'0'

and

'1'

only.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

```
};
```

C#:

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

}
}
```

C:

```
int maxScore(char* s) {

}
```

Go:

```
func maxScore(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

```
}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-score s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int maxScore(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Score After Splitting a String
Difficulty: Easy
Tags: array, string, tree

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function maxScore(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Maximum Score After Splitting a String
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int maxScore(char* s) {

}
```

Go Solution:

```
// Problem: Maximum Score After Splitting a String
// Difficulty: Easy
// Tags: array, string, tree
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maxScore(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Score After Splitting a String
// Difficulty: Easy
// Tags: array, string, tree
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-score s)
(-> string? exact-integer?)
)
```