Problem 3714: Longest Balanced Substring II

Problem Information

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

Problem Description

You are given a string

s

consisting only of the characters

'a'

,

'b'

, and

'c'

.

A

substring

of

s

is called

balanced

if all

distinct

characters in the

substring

appear the

same

number of times.

Return the

length of the longest balanced substring

of

s

.

Example 1:

Input:

s = "abbac"

Output:

4

Explanation:

The longest balanced substring is

"abba"

because both distinct characters

'a'

and

'b'

each appear exactly 2 times.

Example 2:

Input:

s = "aabcc"

Output:

3

Explanation:

The longest balanced substring is

"abc"

because all distinct characters

'a'

,

'b'

and

'c'

each appear exactly 1 time.

Example 3:

Input:

s = "aba"

Output:

2

Explanation:

One of the longest balanced substrings is

"ab"

because both distinct characters

'a'

and

'b'

each appear exactly 1 time. Another longest balanced substring is

"ba"

.

Constraints:

1 <= s.length <= 10

5

s

contains only the characters

'a'

,

'b'

, and

'c'

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int longestBalanced(char* s) {

}
```

Go:

```
func longestBalanced(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

```
*/
function longestBalanced($s) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (longest-balanced s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Balanced Substring II
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 longestBalanced(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Balanced Substring II
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 longestBalanced(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Balanced Substring II
Difficulty: Medium
Tags: array, string, tree, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Balanced Substring II
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 longestBalanced = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Longest Balanced Substring II
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 longestBalanced(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Longest Balanced Substring II
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 LongestBalanced(string s) {

}
}
```

C Solution:

```
/*
* Problem: Longest Balanced Substring II
* Difficulty: Medium
```

```
* Tags: array, string, tree, hash
*
* 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 longestBalanced(char* s) {

}
```

Go Solution:

```
// Problem: Longest Balanced Substring II
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// 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 longestBalanced(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Balanced Substring II
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// 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 longest_balanced(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (longest-balanced s)
(-> string? exact-integer?)
)
```