Problem 1234: Replace the Substring for Balanced String

Problem Information

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

Problem Description

You are given a string s of length

n

containing only four kinds of characters:

'Q'

,

'W'

,

'E'

, and

'R'

.

A string is said to be

balanced

if each of its characters appears

n / 4

times where

n

is the length of the string.

Return

the minimum length of the substring that can be replaced with

any

other string of the same length to make

s

balanced

. If s is already

balanced

, return

0

.

Example 1:

Input:

s = "QWER"

Output:

0

Explanation:

s is already balanced.

Example 2:

Input:

s = "QQWE"

Output:

1

Explanation:

We need to replace a 'Q' to 'R', so that "RQWE" (or "QRWE") is balanced.

Example 3:

Input:

s = "QQQW"

Output:

2

Explanation:

We can replace the first "QQ" to "ER".

Constraints:

n == s.length

4 <= n <= 10

5

n

is a multiple of

4

.

s

contains only

'Q'

,

'W'

,

'E'

, and

'R'

.

Code Snippets

C++:

```
class Solution {
public:
```

```
int balancedString(string s) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int balancedString(char* s) {

}
```

Go:

```
func balancedString(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 balancedString(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 balancedString(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Replace the Substring for Balanced String
Difficulty: Medium
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 balancedString(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 balancedString = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 balancedString(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 BalancedString(string s) {

}
}
```

C Solution:

```
/*
* Problem: Replace the Substring for Balanced String
* Difficulty: Medium
* 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 balancedString(char* s) {

}
```

Go Solution:

```
// Problem: Replace the Substring for Balanced String
// Difficulty: Medium
// 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 balancedString(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Replace the Substring for Balanced String
// Difficulty: Medium
// 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 balanced_string(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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