

# Problem 2609: Find the Longest Balanced Substring of a Binary String

## Problem Information

Difficulty: **Easy**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given a binary string

s

consisting only of zeroes and ones.

A substring of

s

is considered balanced if

all zeroes are before ones

and the number of zeroes is equal to the number of ones inside the substring. Notice that the empty substring is considered a balanced substring.

Return

the length of the longest balanced substring of

s

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

s = "01000111"

Output:

6

Explanation:

The longest balanced substring is "000111", which has length 6.

Example 2:

Input:

s = "00111"

Output:

4

Explanation:

The longest balanced substring is "0011", which has length 4.

Example 3:

Input:

s = "111"

Output:

0

Explanation:

There is no balanced substring except the empty substring, so the answer is 0.

Constraints:

$1 \leq s.length \leq 50$

$'0' \leq s[i] \leq '1'$

## Code Snippets

**C++:**

```
class Solution {
public:
    int findTheLongestBalancedSubstring(string s) {
        }
    };
}
```

**Java:**

```
class Solution {
public int findTheLongestBalancedSubstring(String s) {
        }
    };
}
```

**Python3:**

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

**Python:**

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

### JavaScript:

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

### TypeScript:

```
function findTheLongestBalancedSubstring(s: string): number {  
  
};
```

### C#:

```
public class Solution {  
    public int FindTheLongestBalancedSubstring(string s) {  
  
    }  
}
```

### C:

```
int findTheLongestBalancedSubstring(char* s) {  
  
}
```

### Go:

```
func findTheLongestBalancedSubstring(s string) int {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun findTheLongestBalancedSubstring(s: String): Int {  
  
    }  
}
```

**Swift:**

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

**Rust:**

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

**Ruby:**

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

**PHP:**

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return Integer  
     */  
    function findTheLongestBalancedSubstring($s) {  
  
    }
```

```
}
```

### Dart:

```
class Solution {  
    int findTheLongestBalancedSubstring(String s) {  
  
    }  
}
```

### Scala:

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

### Elixir:

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

### Erlang:

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

### Racket:

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

## Solutions

### C++ Solution:

```
/*
 * Problem: Find the Longest Balanced Substring of a Binary String
 * Difficulty: Easy
 * Tags: string, tree
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

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

    }
};
```

### Java Solution:

```
/**
 * Problem: Find the Longest Balanced Substring of a Binary String
 * Difficulty: Easy
 * Tags: string, tree
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

class Solution {
    public int findTheLongestBalancedSubstring(String s) {

    }
}
```

### Python3 Solution:

```
"""
Problem: Find the Longest Balanced Substring of a Binary String
```

Difficulty: Easy  
Tags: string, tree

Approach: String manipulation with hash map or two pointers  
Time Complexity: O(n) or O(n log n)  
Space Complexity: O(h) for recursion stack where h is height  
"""

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

## Python Solution:

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

## JavaScript Solution:

```
/**  
 * Problem: Find the Longest Balanced Substring of a Binary String  
 * Difficulty: Easy  
 * Tags: string, tree  
 *  
 * Approach: String manipulation with hash map or two pointers  
 * 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 findTheLongestBalancedSubstring = function(s) {  
};
```

### TypeScript Solution:

```
/**  
 * Problem: Find the Longest Balanced Substring of a Binary String  
 * Difficulty: Easy  
 * Tags: string, tree  
 *  
 * Approach: String manipulation with hash map or two pointers  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(h) for recursion stack where h is height  
 */  
  
function findTheLongestBalancedSubstring(s: string): number {  
  
};
```

### C# Solution:

```
/*  
 * Problem: Find the Longest Balanced Substring of a Binary String  
 * Difficulty: Easy  
 * Tags: string, tree  
 *  
 * Approach: String manipulation with hash map or two pointers  
 * 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 FindTheLongestBalancedSubstring(string s) {  
  
    }  
}
```

### C Solution:

```
/*  
 * Problem: Find the Longest Balanced Substring of a Binary String  
 * Difficulty: Easy  
 * Tags: string, tree  
 *  
 * Approach: String manipulation with hash map or two pointers
```

```

* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
int findTheLongestBalancedSubstring(char* s) {
}

```

### Go Solution:

```

// Problem: Find the Longest Balanced Substring of a Binary String
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func findTheLongestBalancedSubstring(s string) int {
}

```

### Kotlin Solution:

```

class Solution {
    fun findTheLongestBalancedSubstring(s: String): Int {
    }
}

```

### Swift Solution:

```

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

```

### Rust Solution:

```

// Problem: Find the Longest Balanced Substring of a Binary String
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

    }
}

```

### Ruby Solution:

```

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

end

```

### PHP Solution:

```

class Solution {

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

    }
}

```

### Dart Solution:

```

class Solution {
    int findTheLongestBalancedSubstring(String s) {
        }

    }
}

```

### **Scala Solution:**

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

### **Elixir Solution:**

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

### **Erlang Solution:**

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

### **Racket Solution:**

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