Problem 1624: Largest Substring Between Two Equal Characters

Problem Information

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

Problem Description

Given a string

s

, return

the length of the longest substring between two equal characters, excluding the two characters.

If there is no such substring return

-1

.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

s = "aa"

Output:

0

Explanation:

The optimal substring here is an empty substring between the two

'a's

.

Example 2:

Input:

s = "abca"

Output:

2

Explanation:

The optimal substring here is "bc".

Example 3:

Input:

s = "cbzxy"

Output:

-1

Explanation:

There are no characters that appear twice in s.

Constraints:

1 <= s.length <= 300

s

contains only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def maxLengthBetweenEqualCharacters(self, s):
"""
:type s: str
```

```
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int maxLengthBetweenEqualCharacters(char* s) {

}
```

Go:

```
func maxLengthBetweenEqualCharacters(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-length-between-equal-characters s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 maxLengthBetweenEqualCharacters(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 maxLengthBetweenEqualCharacters(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Substring Between Two Equal Characters
Difficulty: Easy
Tags: string, tree, hash
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 maxLengthBetweenEqualCharacters = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 maxLengthBetweenEqualCharacters(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 MaxLengthBetweenEqualCharacters(string s) {

}
}
```

C Solution:

```
/*
* Problem: Largest Substring Between Two Equal Characters
* Difficulty: Easy
* Tags: string, tree, hash
*
* 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 maxLengthBetweenEqualCharacters(char* s) {

}
```

Go Solution:

```
// Problem: Largest Substring Between Two Equal Characters
// Difficulty: Easy
// Tags: string, tree, hash
//
// 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 maxLengthBetweenEqualCharacters(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Largest Substring Between Two Equal Characters
// Difficulty: Easy
// Tags: string, tree, hash
```

```
//
// 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 max_length_between_equal_characters(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-length-between-equal-characters s)
(-> string? exact-integer?)
)
```