Problem 3104: Find Longest Self-Contained Substring

Problem Information

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

Problem Description

Given a string

s

, your task is to find the length of the

longest self-contained

substring

of

s

.

A substring

t

of a string

s

is called

self-contained

if

t != s

and for every character in

t

, it doesn't exist in the

rest

of

s

.

Return the length of the

longest

self-contained

substring of

s

if it exists, otherwise, return -1.

Example 1:

Input:

s = "abba"

Output:

2

Explanation:

Let's check the substring

"bb"

. You can see that no other

"b"

is outside of this substring. Hence the answer is 2.

Example 2:

Input:

s = "abab"

Output:

-1

Explanation:

Every substring we choose does not satisfy the described property (there is some character which is inside and outside of that substring). So the answer would be -1.

Example 3:

Input:

s = "abacd"

Output:

4

Explanation:

Let's check the substring

"

abac

"

. There is only one character outside of this substring and that is

"d"

. There is no

"d"

inside the chosen substring, so it satisfies the condition and the answer is 4.

Constraints:

2 <= s.length <= 5 * 10

4

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
```

```
int maxSubstringLength(string s) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int maxSubstringLength(char* s) {

}
```

Go:

```
func maxSubstringLength(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find Longest Self-Contained Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 maxSubstringLength(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Find Longest Self-Contained Substring
```

```
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 maxSubstringLength(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Longest Self-Contained Substring
Difficulty: Hard
Tags: array, string, tree, hash, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Longest Self-Contained Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 maxSubstringLength = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Longest Self-Contained Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 maxSubstringLength(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Find Longest Self-Contained Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 MaxSubstringLength(string s) {

}
}
```

C Solution:

```
/*
* Problem: Find Longest Self-Contained Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 maxSubstringLength(char* s) {

}
```

Go Solution:

```
// Problem: Find Longest Self-Contained Substring
// Difficulty: Hard
// Tags: array, string, tree, hash, search
//
// 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 maxSubstringLength(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Longest Self-Contained Substring
// Difficulty: Hard
// Tags: array, string, tree, hash, search
//
// 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_substring_length(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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