Problem 3: Longest Substring Without Repeating Characters

Problem Information

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

Problem Description

Given a string

s

, find the length of the

longest

substring

without duplicate characters.

Example 1:

Input:

s = "abcabcbb"

Output:

3

Explanation:

The answer is "abc", with the length of 3. Note that

"bca"

and

"cab"

are also correct answers.

Example 2:

Input:

s = "bbbbb"

Output:

1

Explanation:

The answer is "b", with the length of 1.

Example 3:

Input:

s = "pwwkew"

Output:

3

Explanation:

The answer is "wke", with the length of 3. Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Constraints:

0 <= s.length <= 5 * 10

4

s

consists of English letters, digits, symbols and spaces.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int lengthOfLongestSubstring(char* s) {

}
```

Go:

```
func lengthOfLongestSubstring(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
```

```
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Longest Substring Without Repeating Characters
* 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 lengthOfLongestSubstring(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Substring Without Repeating Characters
* 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 lengthOfLongestSubstring(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Substring Without Repeating Characters
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 lengthOfLongestSubstring(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Substring Without Repeating Characters
* 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 lengthOfLongestSubstring = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Substring Without Repeating Characters
* 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 lengthOfLongestSubstring(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Longest Substring Without Repeating Characters
* 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 LengthOfLongestSubstring(string s) {

}
}
```

C Solution:

```
/*
* Problem: Longest Substring Without Repeating Characters
* 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 lengthOfLongestSubstring(char* s) {

}
```

Go Solution:

```
// Problem: Longest Substring Without Repeating Characters
// 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 lengthOfLongestSubstring(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Substring Without Repeating Characters
// 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 length_of_longest_substring(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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