Problem 2414: Length of the Longest Alphabetical Continuous Substring

Problem Information

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

Problem Description

An

alphabetical continuous string

is a string consisting of consecutive letters in the alphabet. In other words, it is any substring of the string

"abcdefghijklmnopqrstuvwxyz"

.

For example,

"abc"

is an alphabetical continuous string, while

"acb"

and

"za"

are not.

Given a string

s

consisting of lowercase letters only, return the

length of the

longest

alphabetical continuous substring.

Example 1:

Input:

s = "abacaba"

Output:

2

Explanation:

There are 4 distinct continuous substrings: "a", "b", "c" and "ab". "ab" is the longest continuous substring.

Example 2:

Input:

s = "abcde"

Output:

5

Explanation:

"abcde" is the longest continuous substring.

Constraints:

1 <= s.length <= 10

5

s

consists of only English lowercase letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def longestContinuousSubstring(self, s):
```

```
"""
:type s: str
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int longestContinuousSubstring(char* s) {

}
```

Go:

```
func longestContinuousSubstring(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 longestContinuousSubstring(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 longestContinuousSubstring(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Length of the Longest Alphabetical Continuous Substring
Difficulty: Medium
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 longestContinuousSubstring(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 longestContinuousSubstring = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 longestContinuousSubstring(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 LongestContinuousSubstring(string s) {

}
}
```

C Solution:

```
/*
* Problem: Length of the Longest Alphabetical Continuous Substring
* Difficulty: Medium
* 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 longestContinuousSubstring(char* s) {

}
```

Go Solution:

```
// Problem: Length of the Longest Alphabetical Continuous Substring
// Difficulty: Medium
// 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 longestContinuousSubstring(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Length of the Longest Alphabetical Continuous Substring
// Difficulty: Medium
// 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 longest_continuous_substring(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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