Problem 1876: Substrings of Size Three with Distinct Characters

Problem Information

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

Problem Description

A string is

good

if there are no repeated characters.

Given a string

s

, return

the number of

good substrings

of length

three

in

s

.

Note that if there are multiple occurrences of the same substring, every occurrence should be counted.

A

substring

is a contiguous sequence of characters in a string.

Example 1:

Input:

s = "xyzzaz"

Output:

1

Explanation:

There are 4 substrings of size 3: "xyz", "yzz", "zza", and "zaz". The only good substring of length 3 is "xyz".

Example 2:

Input:

s = "aababcabc"

Output:

4

Explanation:

There are 7 substrings of size 3: "aab", "aba", "bab", "abc", "bca", "cab", and "abc". The good substrings are "abc", "bca", "cab", and "abc".

Constraints:

1 <= s.length <= 100

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int countGoodSubstrings(char* s) {

}
```

Go:

```
func countGoodSubstrings(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
```

```
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-good-substrings s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 countGoodSubstrings(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 countGoodSubstrings(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Substrings of Size Three with Distinct Characters
Difficulty: Easy
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 countGoodSubstrings(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 countGoodSubstrings = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 countGoodSubstrings(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 CountGoodSubstrings(string s) {

}
}
```

C Solution:

```
/*
* Problem: Substrings of Size Three with Distinct Characters
* Difficulty: Easy
* 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 countGoodSubstrings(char* s) {
```

```
}
```

Go Solution:

```
// Problem: Substrings of Size Three with Distinct Characters
// Difficulty: Easy
// 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 countGoodSubstrings(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Substrings of Size Three with Distinct Characters
// Difficulty: Easy
// 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 count_good_substrings(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

```
}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-good-substrings s)
(-> string? exact-integer?)
)
```