Problem 1358: Number of Substrings Containing All Three Characters

Problem Information

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

Problem Description

Given a string

s

consisting only of characters

a

,

b

and

c

.

Return the number of substrings containing

at least

one occurrence of all these characters

a

,

b

and

c

.

Example 1:

Input:

s = "abcabc"

Output:

10

Explanation:

The substrings containing at least one occurrence of the characters

a

,

b

and

c are "

abc

", "

abca

", "

abcab

", "

abcabc

", "

bca

", "

bcab

", "

bcabc

", "

cab

", "

cabc

"

and

"

abc

"

(

again

)

.

Example 2:

Input:

s = "aaacb"

Output:

3

Explanation:

The substrings containing at least one occurrence of the characters

a

,

b

and

c are "

aaacb

", "

aacb

"

and

"

acb

".

Example 3:

Input:

s = "abc"

Output:

1

Constraints:

3 <= s.length <= 5 x 10^4

s

only consists of

a

,

b

or

c

characters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int numberOfSubstrings(char* s) {

}
```

Go:

```
func numberOfSubstrings(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
```

```
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (number-of-substrings s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Substrings Containing All Three 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 numberOfSubstrings = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(s: string): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Number of Substrings Containing All Three 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 NumberOfSubstrings(string s) {

}
}
```

C Solution:

```
/*
* Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(char* s) {

}
```

Go Solution:

```
// Problem: Number of Substrings Containing All Three 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 numberOfSubstrings(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Substrings Containing All Three 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 number_of_substrings(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (number-of-substrings s)
(-> string? exact-integer?)
)
```