

Problem 2067: Number of Equal Count Substrings

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a

0-indexed

string

s

consisting of only lowercase English letters, and an integer

count

. A

substring

of

s

is said to be an

equal count substring

if, for each

unique

letter in the substring, it appears exactly

count

times in the substring.

Return

the number of

equal count substrings

in

s

.

A

substring

is a contiguous non-empty sequence of characters within a string.

Example 1:

Input:

s = "aaabcbcc", count = 3

Output:

3

Explanation:

The substring that starts at index 0 and ends at index 2 is "aaa". The letter 'a' in the substring appears exactly 3 times. The substring that starts at index 3 and ends at index 8 is "bcbbcc". The letters 'b' and 'c' in the substring appear exactly 3 times. The substring that starts at index 0 and ends at index 8 is "aaabcbbcc". The letters 'a', 'b', and 'c' in the substring appear exactly 3 times.

Example 2:

Input:

s = "abcd", count = 2

Output:

0

Explanation:

The number of times each letter appears in s is less than count. Therefore, no substrings in s are equal count substrings, so return 0.

Example 3:

Input:

s = "a", count = 5

Output:

0

Explanation:

The number of times each letter appears in s is less than count. Therefore, no substrings in s are equal count substrings, so return 0

Constraints:

$1 \leq s.length \leq 3 * 10$

4

1 <= count <= 3 * 10

4

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
    int equalCountSubstrings(string s, int count) {

    }
};
```

Java:

```
class Solution {
    public int equalCountSubstrings(String s, int count) {

    }
}
```

Python3:

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

Python:

```
class Solution(object):
    def equalCountSubstrings(self, s, count):
        """
        :type s: str
```

```
:type count: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function equalCountSubstrings(s: string, count: number): number {

};
```

C#:

```
public class Solution {
    public int EqualCountSubstrings(string s, int count) {

    }
}
```

C:

```
int equalCountSubstrings(char* s, int count) {

}
```

Go:

```
func equalCountSubstrings(s string, count int) int {

}
```

Kotlin:

```

class Solution {
    fun equalCountSubstrings(s: String, count: Int): Int {

    }
}

```

Swift:

```

class Solution {
    func equalCountSubstrings(_ s: String, _ count: Int) -> Int {

    }
}

```

Rust:

```

impl Solution {
    pub fn equal_count_substrings(s: String, count: i32) -> i32 {

    }
}

```

Ruby:

```

# @param {String} s
# @param {Integer} count
# @return {Integer}
def equal_count_substrings(s, count)

end

```

PHP:

```

class Solution {

    /**
     * @param String $s
     * @param Integer $count
     * @return Integer
     */
    function equalCountSubstrings($s, $count) {

    }
}

```

```
}
```

Dart:

```
class Solution {  
  int equalCountSubstrings(String s, int count) {  
  
  }  
}
```

Scala:

```
object Solution {  
  def equalCountSubstrings(s: String, count: Int): Int = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec equal_count_substrings(s :: String.t, count :: integer) :: integer  
  def equal_count_substrings(s, count) do  
  
  end  
end
```

Erlang:

```
-spec equal_count_substrings(S :: unicode:unicode_binary(), Count ::  
integer()) -> integer().  
equal_count_substrings(S, Count) ->  
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
 * Problem: Number of Equal Count Substrings
 * 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 equalCountSubstrings(string s, int count) {

    }
};
```

Java Solution:

```
/**
 * Problem: Number of Equal Count Substrings
 * 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 equalCountSubstrings(String s, int count) {

    }
}
```

Python3 Solution:

```
"""
Problem: Number of Equal Count Substrings
```

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 equalCountSubstrings(self, s: str, count: int) -> int:
```

```
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
```

```
def equalCountSubstrings(self, s, count):
```

```
"""
```

```
:type s: str
```

```
:type count: int
```

```
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
```

```
 * Problem: Number of Equal Count Substrings
```

```
 * 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
```

```
 * @param {number} count
```

```
 * @return {number}
```

```
 */
```

```
var equalCountSubstrings = function(s, count) {
```

```
};
```

TypeScript Solution:

```
/**
 * Problem: Number of Equal Count Substrings
 * 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 equalCountSubstrings(s: string, count: number): number {

};
```

C# Solution:

```
/*
 * Problem: Number of Equal Count Substrings
 * 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 EqualCountSubstrings(string s, int count) {

    }
}
```

C Solution:

```
/*
 * Problem: Number of Equal Count Substrings
```

```

* 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 equalCountSubstrings(char* s, int count) {

}

```

Go Solution:

```

// Problem: Number of Equal Count Substrings
// 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 equalCountSubstrings(s string, count int) int {

}

```

Kotlin Solution:

```

class Solution {
    fun equalCountSubstrings(s: String, count: Int): Int {

    }
}

```

Swift Solution:

```

class Solution {
    func equalCountSubstrings(_ s: String, _ count: Int) -> Int {

    }
}

```

Rust Solution:

```
// Problem: Number of Equal Count Substrings
// 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 equal_count_substrings(s: String, count: i32) -> i32 {

    }
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} count
# @return {Integer}
def equal_count_substrings(s, count)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param String $s
     * @param Integer $count
     * @return Integer
     */
    function equalCountSubstrings($s, $count) {

    }

}
```

Dart Solution:

```

class Solution {
  int equalCountSubstrings(String s, int count) {

  }
}

```

Scala Solution:

```

object Solution {
  def equalCountSubstrings(s: String, count: Int): Int = {

  }
}

```

Elixir Solution:

```

defmodule Solution do
  @spec equal_count_substrings(s :: String.t, count :: integer) :: integer
  def equal_count_substrings(s, count) do

  end
end

```

Erlang Solution:

```

-spec equal_count_substrings(S :: unicode:unicode_binary(), Count ::
integer()) -> integer().
equal_count_substrings(S, Count) ->
.

```

Racket Solution:

```

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

```