Problem 2168: Unique Substrings With Equal Digit Frequency

Problem Information

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

Problem Description

Given a digit string

s

, return

the number of

unique substrings

of

s

where every digit appears the same number of times.

Example 1:

Input:

s = "1212"

Output:

5

Explanation:

The substrings that meet the requirements are "1", "2", "12", "21", "1212". Note that although the substring "12" appears twice, it is only counted once.

Example 2:

Input:

s = "12321"

Output:

9

Explanation:

The substrings that meet the requirements are "1", "2", "3", "12", "23", "32", "21", "123", "321".

Constraints:

1 <= s.length <= 1000

s

consists of digits.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int EqualDigitFrequency(string s) {
}
}
```

C:

```
int equalDigitFrequency(char* s) {

}
```

Go:

```
func equalDigitFrequency(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (equal-digit-frequency s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 equalDigitFrequency(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 equalDigitFrequency(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Unique Substrings With Equal Digit Frequency
Difficulty: Medium
Tags: string, tree, hash

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 equalDigitFrequency(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 equalDigitFrequency = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 equalDigitFrequency(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 EqualDigitFrequency(string s) {

}
}
```

C Solution:

```
/*
* Problem: Unique Substrings With Equal Digit Frequency
* Difficulty: Medium
* Tags: string, tree, hash
*
* 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 equalDigitFrequency(char* s) {

}
```

Go Solution:

```
// Problem: Unique Substrings With Equal Digit Frequency
// Difficulty: Medium
// Tags: string, tree, hash
//
// 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 equalDigitFrequency(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Unique Substrings With Equal Digit Frequency
// Difficulty: Medium
// Tags: string, tree, hash
//
// 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 equal_digit_frequency(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (equal-digit-frequency s)
(-> string? exact-integer?)
)
```