

Problem 2743: Count Substrings Without Repeating Character

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a string

`s`

consisting only of lowercase English letters. We call a substring

special

if it contains no character which has occurred at least twice (in other words, it does not contain a repeating character). Your task is to count the number of

special

substrings. For example, in the string

`"pop"`

, the substring

`"po"`

is a

special

substring, however,

"pop"

is not

special

(since

'p'

has occurred twice).

Return

the number of

special

substrings.

A

substring

is a contiguous sequence of characters within a string. For example,

"abc"

is a substring of

"abcd"

, but

"acd"

is not.

Example 1:

Input:

`s = "abcd"`

Output:

10

Explanation:

Since each character occurs once, every substring is a special substring. We have 4 substrings of length one, 3 of length two, 2 of length three, and 1 substring of length four. So overall there are $4 + 3 + 2 + 1 = 10$ special substrings.

Example 2:

Input:

`s = "ooo"`

Output:

3

Explanation:

Any substring with a length of at least two contains a repeating character. So we have to count the number of substrings of length one, which is 3.

Example 3:

Input:

`s = "abab"`

Output:

7

Explanation:

Special substrings are as follows (sorted by their start positions): Special substrings of length 1: "a", "b", "a", "b" Special substrings of length 2: "ab", "ba", "ab" And it can be shown that there are no special substrings with a length of at least three. So the answer would be $4 + 3 = 7$.

Constraints:

$1 \leq s.length \leq 10$

s

s

consists of lowercase English letters

Code Snippets

C++:

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

    }
};
```

Java:

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

    }
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

    }
}
```

C:

```
int numberOfSpecialSubstrings(char* s) {

}
```

Go:

```
func numberOfSpecialSubstrings(s string) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun numberOfSpecialSubstrings(s: String): Int {  
  
    }  
}
```

Swift:

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

Rust:

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

Ruby:

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

PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return Integer  
     */  
}
```

```

*/
function numberOfSpecialSubstrings($s) {

}

}

```

Dart:

```

class Solution {
  int numberOfSpecialSubstrings(String s) {

  }

}

```

Scala:

```

object Solution {
  def numberOfSpecialSubstrings(s: String): Int = {

  }

}

```

Elixir:

```

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

  end

end

```

Erlang:

```

-spec number_of_special_substrings(S :: unicode:unicode_binary()) ->
integer().
number_of_special_substrings(S) ->
.

```

Racket:

```

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

```

```
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Count Substrings Without Repeating Character
 * Difficulty: Medium
 * Tags: array, string, tree, hash, sort
 *
 * 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 numberOfSpecialSubstrings(string s) {

    }
};
```

Java Solution:

```
/**
 * Problem: Count Substrings Without Repeating Character
 * Difficulty: Medium
 * Tags: array, string, tree, hash, sort
 *
 * 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 numberOfSpecialSubstrings(String s) {

    }
}
```

Python3 Solution:

```
"""
Problem: Count Substrings Without Repeating Character
Difficulty: Medium
Tags: array, string, tree, hash, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
 * Problem: Count Substrings Without Repeating Character
 * Difficulty: Medium
 * Tags: array, string, tree, hash, sort
 *
 * 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 numberOfSpecialSubstrings = function(s) {  
  
};
```

TypeScript Solution:

```
/**  
 * Problem: Count Substrings Without Repeating Character  
 * Difficulty: Medium  
 * Tags: array, string, tree, hash, sort  
 *  
 * 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 numberOfSpecialSubstrings(s: string): number {  
  
};
```

C# Solution:

```
/*  
 * Problem: Count Substrings Without Repeating Character  
 * Difficulty: Medium  
 * Tags: array, string, tree, hash, sort  
 *  
 * 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 NumberOfSpecialSubstrings(string s) {  
  
    }  
}
```

C Solution:

```

/*
 * Problem: Count Substrings Without Repeating Character
 * Difficulty: Medium
 * Tags: array, string, tree, hash, sort
 *
 * 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 numberOfSpecialSubstrings(char* s) {

}

```

Go Solution:

```

// Problem: Count Substrings Without Repeating Character
// Difficulty: Medium
// Tags: array, string, tree, hash, sort
//
// 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 numberOfSpecialSubstrings(s string) int {

}

```

Kotlin Solution:

```

class Solution {
    fun numberOfSpecialSubstrings(s: String): Int {

    }
}

```

Swift Solution:

```

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

    }
}

```

```
}
```

Rust Solution:

```
// Problem: Count Substrings Without Repeating Character
// Difficulty: Medium
// Tags: array, string, tree, hash, sort
//
// 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_special_substrings(s: String) -> i32 {

    }
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

    }

}
```

Dart Solution:

```
class Solution {  
  int numberOfSpecialSubstrings(String s) {  
  
  }  
}
```

Scala Solution:

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

Elixir Solution:

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

Erlang Solution:

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

Racket Solution:

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