

Problem 3306: Count of Substrings Containing Every Vowel and K Consonants II

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a string

word

and a

non-negative

integer

k

.

Return the total number of

substrings

of

word

that contain every vowel (

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

)

at least

once and

exactly

k

consonants.

Example 1:

Input:

word = "aeioqq", k = 1

Output:

0

Explanation:

There is no substring with every vowel.

Example 2:

Input:

word = "aeiou", k = 0

Output:

1

Explanation:

The only substring with every vowel and zero consonants is

word[0..4]

, which is

"aeiou"

Example 3:

Input:

word = "

ieaouqqieaouqq

", k = 1

Output:

Explanation:

The substrings with every vowel and one consonant are:

word[0..5]

, which is

"ieaouq"

word[6..11]

, which is

"qieaou"

word[7..12]

, which is

"ieaouq"

Constraints:

$5 \leq \text{word.length} \leq 2 * 10^5$

5

word

consists only of lowercase English letters.

$0 \leq k \leq \text{word.length} - 5$

Code Snippets

C++:

```
class Solution {  
public:  
    long long countOfSubstrings(string word, int k) {  
  
    }  
};
```

Java:

```
class Solution {  
public long countOfSubstrings(String word, int k) {  
  
}  
}
```

Python3:

```
class Solution:  
    def countOfSubstrings(self, word: str, k: int) -> int:
```

Python:

```
class Solution(object):  
    def countOfSubstrings(self, word, k):  
        """  
        :type word: str  
        :type k: int  
        :rtype: int  
        """
```

JavaScript:

```
/**  
 * @param {string} word  
 * @param {number} k
```

```
* @return {number}
*/
var countOfSubstrings = function(word, k) {

};
```

TypeScript:

```
function countOfSubstrings(word: string, k: number): number {
}
```

C#:

```
public class Solution {
    public long CountOfSubstrings(string word, int k) {
        }
}
```

C:

```
long long countOfSubstrings(char* word, int k) {
}
```

Go:

```
func countOfSubstrings(word string, k int) int64 {
}
```

Kotlin:

```
class Solution {
    fun countOfSubstrings(word: String, k: Int): Long {
    }
}
```

Swift:

```
class Solution {  
    func countOfSubstrings(_ word: String, _ k: Int) -> Int {  
        }  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn count_of_substrings(word: String, k: i32) -> i64 {  
        }  
    }  
}
```

Ruby:

```
# @param {String} word  
# @param {Integer} k  
# @return {Integer}  
def count_of_substrings(word, k)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param String $word  
     * @param Integer $k  
     * @return Integer  
     */  
    function countOfSubstrings($word, $k) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int countOfSubstrings(String word, int k) {  
    }  
}
```

```
}
```

Scala:

```
object Solution {  
    def countOfSubstrings(word: String, k: Int): Long = {  
        }  
    }  
}
```

Elixir:

```
defmodule Solution do  
    @spec count_of_substrings(word :: String.t, k :: integer) :: integer  
    def count_of_substrings(word, k) do  
  
    end  
    end
```

Erlang:

```
-spec count_of_substrings(Word :: unicode:unicode_binary(), K :: integer())  
-> integer().  
count_of_substrings(Word, K) ->  
.
```

Racket:

```
(define/contract (count-of-substrings word k)  
  (-> string? exact-integer? exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
* Problem: Count of Substrings Containing Every Vowel and K Consonants II  
* 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:
long long countOfSubstrings(string word, int k) {

}
};


```

Java Solution:

```

/**
* Problem: Count of Substrings Containing Every Vowel and K Consonants II
* 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 long countOfSubstrings(String word, int k) {

}
};


```

Python3 Solution:

```

"""
Problem: Count of Substrings Containing Every Vowel and K Consonants II
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 countOfSubstrings(self, word: str, k: int) -> int:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):
    def countOfSubstrings(self, word, k):
        """
        :type word: str
        :type k: int
        :rtype: int
        """
```

JavaScript Solution:

```
/**
 * Problem: Count of Substrings Containing Every Vowel and K Consonants II
 * 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} word
 * @param {number} k
 * @return {number}
 */
var countOfSubstrings = function(word, k) {

};
```

TypeScript Solution:

```

/**
 * Problem: Count of Substrings Containing Every Vowel and K Consonants II
 * 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 countOfSubstrings(word: string, k: number): number {
}

```

C# Solution:

```

/*
 * Problem: Count of Substrings Containing Every Vowel and K Consonants II
 * 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 long CountOfSubstrings(string word, int k) {
}
}

```

C Solution:

```

/*
 * Problem: Count of Substrings Containing Every Vowel and K Consonants II
 * 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

```

```
*/  
  
long long countOfSubstrings(char* word, int k) {  
  
}
```

Go Solution:

```
// Problem: Count of Substrings Containing Every Vowel and K Consonants II  
// 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 countOfSubstrings(word string, k int) int64 {  
  
}
```

Kotlin Solution:

```
class Solution {  
    fun countOfSubstrings(word: String, k: Int): Long {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func countOfSubstrings(_ word: String, _ k: Int) -> Int {  
  
    }  
}
```

Rust Solution:

```
// Problem: Count of Substrings Containing Every Vowel and K Consonants II  
// 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 count_of_substrings(word: String, k: i32) -> i64 {

}
}

```

Ruby Solution:

```

# @param {String} word
# @param {Integer} k
# @return {Integer}
def count_of_substrings(word, k)

end

```

PHP Solution:

```

class Solution {

/**
 * @param String $word
 * @param Integer $k
 * @return Integer
 */
function countOfSubstrings($word, $k) {

}
}

```

Dart Solution:

```

class Solution {
int countOfSubstrings(String word, int k) {

}
}

```

Scala Solution:

```
object Solution {  
    def countOfSubstrings(word: String, k: Int): Long = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec count_of_substrings(word :: String.t, k :: integer) :: integer  
  def count_of_substrings(word, k) do  
  
  end  
end
```

Erlang Solution:

```
-spec count_of_substrings(Word :: unicode:unicode_binary(), K :: integer())  
-> integer().  
count_of_substrings(Word, K) ->  
.
```

Racket Solution:

```
(define/contract (count-of-substrings word k)  
  (-> string? exact-integer? exact-integer?)  
)
```