# Problem 3325: Count Substrings With K-Frequency Characters I

## Problem Information

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

## Problem Description

Given a string

s

and an integer

k

, return the total number of

substrings

of

s

where

at least one

character appears

at least

k

times.

Example 1:

Input:

s = "abacb", k = 2

Output:

4

Explanation:

The valid substrings are:

"aba"

(character

'a'

appears 2 times).

"abac"

(character

'a'

appears 2 times).

"abacb"

(character

'a'

appears 2 times).

"bacb"

(character

'b'

appears 2 times).

Example 2:

Input:

s = "abcde", k = 1

Output:

15

Explanation:

All substrings are valid because every character appears at least once.

Constraints:

1 <= s.length <= 3000

1 <= k <= s.length

s

consists only of lowercase English letters.

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
int numberOfSubstrings(char* s, int k) {

}
```

**Go:**

```
func numberOfSubstrings(s string, k int) int {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
}
```

**Scala:**

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

}
```

```
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

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

## Solutions

### C++ Solution:

```
/*
* Problem: Count Substrings With K-Frequency Characters I
* 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, int k) {

}
};
```

**Java Solution:**

```
/**
* Problem: Count Substrings With K-Frequency Characters I
* 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, int k) {

}
}
```

**Python3 Solution:**

```
"""
Problem: Count Substrings With K-Frequency Characters I
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, k: int) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

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

### JavaScript Solution:

```
/**
* Problem: Count Substrings With K-Frequency Characters I
* 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} k
* @return {number}
*/
var numberOfSubstrings = function(s, k) {

};
```

### TypeScript Solution:

```
/**
* Problem: Count Substrings With K-Frequency Characters I
* 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, k: number): number {
```

```
};
```

### C# Solution:

```
/*
* Problem: Count Substrings With K-Frequency Characters I
* 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, int k) {

}
}
```

### C Solution:

```
/*
* Problem: Count Substrings With K-Frequency Characters I
* 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, int k) {

}
```

### Go Solution:

```
// Problem: Count Substrings With K-Frequency Characters I
// 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, k int) int {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Count Substrings With K-Frequency Characters I
// 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, k: i32) -> i32 {

}
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

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

}
}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

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