Problem 1297: Maximum Number of Occurrences of a Substring

Problem Information

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

Problem Description

Given a string

s

, return the maximum number of occurrences of

any

substring under the following rules:

The number of unique characters in the substring must be less than or equal to

maxLetters

.

The substring size must be between

minSize

and

maxSize

inclusive.

Example 1:

Input:

s = "aababcaab", maxLetters = 2, minSize = 3, maxSize = 4

Output:

2

Explanation:

Substring "aab" has 2 occurrences in the original string. It satisfies the conditions, 2 unique letters and size 3 (between minSize and maxSize).

Example 2:

Input:

s = "aaaa", maxLetters = 1, minSize = 3, maxSize = 3

Output:

2

Explanation:

Substring "aaa" occur 2 times in the string. It can overlap.

Constraints:

1 <= s.length <= 10

5

1 <= maxLetters <= 26

1 <= minSize <= maxSize <= min(26, s.length)

s

consists of only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int maxFreq(string s, int maxLetters, int minSize, int maxSize) {

}
};
```

Java:

```
class Solution {
public int maxFreq(String s, int maxLetters, int minSize, int maxSize) {

}
}
```

Python3:

```
class Solution:
def maxFreq(self, s: str, maxLetters: int, minSize: int, maxSize: int) ->
int:
```

Python:

```
class Solution(object):
def maxFreq(self, s, maxLetters, minSize, maxSize):
"""
:type s: str
:type maxLetters: int
:type minSize: int
:type maxSize: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} maxLetters
* @param {number} minSize
* @param {number} maxSize
* @return {number}
*/
var maxFreq = function(s, maxLetters, minSize, maxSize) {

};
```

TypeScript:

```
function maxFreq(s: string, maxLetters: number, minSize: number, maxSize:
number): number {

};
```

C#:

```
public class Solution {
public int MaxFreq(string s, int maxLetters, int minSize, int maxSize) {

}
}
```

C:

```
int maxFreq(char* s, int maxLetters, int minSize, int maxSize) {

}
```

Go:

```
func maxFreq(s string, maxLetters int, minSize int, maxSize int) int {

}
```

Kotlin:

```
class Solution {
fun maxFreq(s: String, maxLetters: Int, minSize: Int, maxSize: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxFreq(_ s: String, _ maxLetters: Int, _ minSize: Int, _ maxSize: Int)
-> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_freq(s: String, max_letters: i32, min_size: i32, max_size: i32) ->
i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} max_letters
# @param {Integer} min_size
# @param {Integer} max_size
# @return {Integer}
def max_freq(s, max_letters, min_size, max_size)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $maxLetters
* @param Integer $minSize
```

```
* @param Integer $maxSize
* @return Integer
*/
function maxFreq($s, $maxLetters, $minSize, $maxSize) {

}
}
```

Dart:

```
class Solution {
int maxFreq(String s, int maxLetters, int minSize, int maxSize) {

}
}
```

Scala:

```
object Solution {
def maxFreq(s: String, maxLetters: Int, minSize: Int, maxSize: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_freq(s :: String.t, max_letters :: integer, min_size :: integer,
max_size :: integer) :: integer
def max_freq(s, max_letters, min_size, max_size) do

end
end
```

Erlang:

```
-spec max_freq(S :: unicode:unicode_binary(), MaxLetters :: integer(),
MinSize :: integer(), MaxSize :: integer()) -> integer().
max_freq(S, MaxLetters, MinSize, MaxSize) ->
.
```

Racket:

```
(define/contract (max-freq s maxLetters minSize maxSize)
(-> string? exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Occurrences of a Substring
* 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 maxFreq(string s, int maxLetters, int minSize, int maxSize) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Occurrences of a Substring
* 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 maxFreq(String s, int maxLetters, int minSize, int maxSize) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Occurrences of a Substring
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 maxFreq(self, s: str, maxLetters: int, minSize: int, maxSize: int) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxFreq(self, s, maxLetters, minSize, maxSize):
"""
:type s: str
:type maxLetters: int
:type minSize: int
:type maxSize: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Number of Occurrences of a Substring
* 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} maxLetters
* @param {number} minSize
* @param {number} maxSize
* @return {number}
*/
var maxFreq = function(s, maxLetters, minSize, maxSize) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Occurrences of a Substring
* 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 maxFreq(s: string, maxLetters: number, minSize: number, maxSize:
number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Occurrences of a Substring
* 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 MaxFreq(string s, int maxLetters, int minSize, int maxSize) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Occurrences of a Substring
* 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 maxFreq(char* s, int maxLetters, int minSize, int maxSize) {

}
```

Go Solution:

```
// Problem: Maximum Number of Occurrences of a Substring
// 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 maxFreq(s string, maxLetters int, minSize int, maxSize int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxFreq(s: String, maxLetters: Int, minSize: Int, maxSize: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxFreq(_ s: String, _ maxLetters: Int, _ minSize: Int, _ maxSize: Int)
-> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Occurrences of a Substring
// 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 max_freq(s: String, max_letters: i32, min_size: i32, max_size: i32) ->
i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} max_letters
# @param {Integer} min_size
# @param {Integer} max_size
# @return {Integer}
def max_freq(s, max_letters, min_size, max_size)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $maxLetters
* @param Integer $minSize
* @param Integer $maxSize
* @return Integer
*/
function maxFreq($s, $maxLetters, $minSize, $maxSize) {

}
}
```

Dart Solution:

```
class Solution {
int maxFreq(String s, int maxLetters, int minSize, int maxSize) {

}
}
```

Scala Solution:

```
object Solution {
def maxFreq(s: String, maxLetters: Int, minSize: Int, maxSize: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_freq(s :: String.t, max_letters :: integer, min_size :: integer,
max_size :: integer) :: integer
def max_freq(s, max_letters, min_size, max_size) do

end
end
```

Erlang Solution:

```
-spec max_freq(S :: unicode:unicode_binary(), MaxLetters :: integer(),
MinSize :: integer(), MaxSize :: integer()) -> integer().
max_freq(S, MaxLetters, MinSize, MaxSize) ->
.
```

Racket Solution:

```
(define/contract (max-freq s maxLetters minSize maxSize)
(-> string? exact-integer? exact-integer? exact-integer? exact-integer?)
)
```