Problem 340: Longest Substring with At Most K Distinct Characters

Problem Information

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

Problem Description

Given a string

s

and an integer

k

, return

the length of the longest

substring

of

s

that contains at most

k

distinct

characters

.

Example 1:

Input:

s = "eceba", k = 2

Output:

3

Explanation:

The substring is "ece" with length 3.

Example 2:

Input:

s = "aa", k = 1

Output:

2

Explanation:

The substring is "aa" with length 2.

Constraints:

1 <= s.length <= 5 * 10

4

0 <= k <= 50

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (length-of-longest-substring-k-distinct s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Substring with At Most K Distinct Characters
* 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 lengthOfLongestSubstringKDistinct(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Substring with At Most K Distinct Characters
* 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 lengthOfLongestSubstringKDistinct(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Substring with At Most K Distinct Characters
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 lengthOfLongestSubstringKDistinct(self, s: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Substring with At Most K Distinct Characters
* 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 lengthOfLongestSubstringKDistinct = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Substring with At Most K Distinct Characters
* 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 lengthOfLongestSubstringKDistinct(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Longest Substring with At Most K Distinct Characters
* 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 LengthOfLongestSubstringKDistinct(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Longest Substring with At Most K Distinct Characters
* 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 lengthOfLongestSubstringKDistinct(char* s, int k) {
}
```

Go Solution:

```
// Problem: Longest Substring with At Most K Distinct Characters
// 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 lengthOfLongestSubstringKDistinct(s string, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Substring with At Most K Distinct Characters
// 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 length_of_longest_substring_k_distinct(s: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (length-of-longest-substring-k-distinct s k)
(-> string? exact-integer? exact-integer?)
)
```