Problem 3090: Maximum Length Substring With Two Occurrences

Problem Information

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

Problem Description

Given a string

s

, return the

maximum

length of a

substring

such that it contains

at most two occurrences

of each character.

Example 1:

Input:

s = "bcbbbcba"

Output:

4

Explanation:

The following substring has a length of 4 and contains at most two occurrences of each character:

"bcbb

bcba

"

.

Example 2:

Input:

s = "aaaa"

Output:

2

Explanation:

The following substring has a length of 2 and contains at most two occurrences of each character:

"

aa

aa"

.

Constraints:

2 <= s.length <= 100

s

consists only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int maximumLengthSubstring(char* s) {

}
```

Go:

```
func maximumLengthSubstring(s string) int {

}
```

Kotlin:

```
class Solution {
fun maximumLengthSubstring(s: String): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int maximumLengthSubstring(String s) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-length-substring s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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 maximumLengthSubstring(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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 maximumLengthSubstring(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Length Substring With Two Occurrences
Difficulty: Easy
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 maximumLengthSubstring(self, s: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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
* @return {number}
*/
var maximumLengthSubstring = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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 maximumLengthSubstring(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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 MaximumLengthSubstring(string s) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Length Substring With Two Occurrences
* Difficulty: Easy
* 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 maximumLengthSubstring(char* s) {

}
```

Go Solution:

```
// Problem: Maximum Length Substring With Two Occurrences
// Difficulty: Easy
// 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 maximumLengthSubstring(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumLengthSubstring(s: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Length Substring With Two Occurrences
// Difficulty: Easy
// 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 maximum_length_substring(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int maximumLengthSubstring(String s) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-length-substring s)
(-> string? exact-integer?)
)
```