Problem 3399: Smallest Substring With Identical Characters II

Problem Information

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

Problem Description

You are given a binary string

s

of length

n

and an integer

numOps

.

You are allowed to perform the following operation on

s

at most

numOps

times:

Select any index

i

(where

0 <= i < n

) and

flip

s[i]

. If

s[i] == '1'

, change

s[i]

to

'0'

and vice versa.

You need to

minimize

the length of the

longest

substring

of

s

such that all the characters in the substring are

identical

.

Return the

minimum

length after the operations.

Example 1:

Input:

s = "000001", numOps = 1

Output:

2

Explanation:

By changing

s[2]

to

'1'

,

s

becomes

"001001"

. The longest substrings with identical characters are

s[0..1]

and

s[3..4]

.

Example 2:

Input:

s = "0000", numOps = 2

Output:

1

Explanation:

By changing

s[0]

and

s[2]

to

'1'

,

s

becomes

"1010"

.

Example 3:

Input:

s = "0101", numOps = 0

Output:

1

Constraints:

1 <= n == s.length <= 10

5

s

consists only of

'0'

and

'1'

.

0 <= numOps <= n

Code Snippets

C++:

```
class Solution {
public:
int minLength(string s, int numOps) {

}
};
```

Java:

```
class Solution {
public int minLength(String s, int numOps) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
};
```

TypeScript:

```
function minLength(s: string, numOps: number): number {

};
```

C#:

```
public class Solution {
public int MinLength(string s, int numOps) {

}
}
```

C:

```
int minLength(char* s, int numOps) {

}
```

Go:

```
func minLength(s string, numOps int) int {

}
```

Kotlin:

```
class Solution {
fun minLength(s: String, numOps: Int): Int {

}
}
```

Swift:

```
class Solution {
func minLength(_ s: String, _ numOps: Int) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn min_length(s: String, num_ops: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} num_ops
# @return {Integer}
def min_length(s, num_ops)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $numOps
* @return Integer
*/
function minLength($s, $numOps) {

}
}
```

Dart:

```
class Solution {
int minLength(String s, int numOps) {

}
}
```

Scala:

```
object Solution {
def minLength(s: String, numOps: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec min_length(S :: unicode:unicode_binary(), NumOps :: integer()) ->
integer().
min_length(S, NumOps) ->
.
```

Racket:

```
(define/contract (min-length s numOps)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public:
int minLength(string s, int numOps) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int minLength(String s, int numOps) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Substring With Identical Characters II
Difficulty: Hard
Tags: string, tree, search

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def minLength(self, s: str, numOps: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
* 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} numOps
* @return {number}
*/
var minLength = function(s, numOps) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function minLength(s: string, numOps: number): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
* 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 MinLength(string s, int numOps) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Substring With Identical Characters II
* Difficulty: Hard
* Tags: string, tree, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int minLength(char* s, int numOps) {

}
```

Go Solution:

```
// Problem: Smallest Substring With Identical Characters II
// Difficulty: Hard
// Tags: string, tree, search
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func minLength(s string, numOps int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minLength(s: String, numOps: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minLength(_ s: String, _ numOps: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Smallest Substring With Identical Characters II
// Difficulty: Hard
// Tags: string, tree, search
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn min_length(s: String, num_ops: i32) -> i32 {
```

```
}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} num_ops
# @return {Integer}
def min_length(s, num_ops)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $numOps
* @return Integer
*/
function minLength($s, $numOps) {

}
}
```

Dart Solution:

```
class Solution {
int minLength(String s, int numOps) {

}
}
```

Scala Solution:

```
object Solution {
def minLength(s: String, numOps: Int): Int = {

}
```

```
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec min_length(S :: unicode:unicode_binary(), NumOps :: integer()) ->
integer().
min_length(S, NumOps) ->
.
```

Racket Solution:

```
(define/contract (min-length s numOps)
(-> string? exact-integer? exact-integer?)
)
```