

Problem 1750: Minimum Length of String After Deleting Similar Ends

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a string

s

consisting only of characters

'a'

,

'b'

, and

'c'

. You are asked to apply the following algorithm on the string any number of times:

Pick a

non-empty

prefix from the string

s

where all the characters in the prefix are equal.

Pick a

non-empty

suffix from the string

s

where all the characters in this suffix are equal.

The prefix and the suffix should not intersect at any index.

The characters from the prefix and suffix must be the same.

Delete both the prefix and the suffix.

Return

the

minimum length

of

s

after performing the above operation any number of times (possibly zero times)

Example 1:

Input:

s = "ca"

Output:

2

Explanation:

You can't remove any characters, so the string stays as is.

Example 2:

Input:

s = "cabaabac"

Output:

0

Explanation:

An optimal sequence of operations is: - Take prefix = "c" and suffix = "c" and remove them, s = "abaaba". - Take prefix = "a" and suffix = "a" and remove them, s = "baab". - Take prefix = "b" and suffix = "b" and remove them, s = "aa". - Take prefix = "a" and suffix = "a" and remove them, s = "".

Example 3:

Input:

s = "aabccabba"

Output:

3

Explanation:

An optimal sequence of operations is: - Take prefix = "aa" and suffix = "a" and remove them, s = "bccabb". - Take prefix = "b" and suffix = "bb" and remove them, s = "cca".

Constraints:

$1 \leq s.length \leq 10$

5

s

only consists of characters

'a'

,

'b'

, and

'c'

.

Code Snippets

C++:

```
class Solution {
public:
    int minimumLength(string s) {
        }
    };
}
```

Java:

```
class Solution {
    public int minimumLength(String s) {
        }
}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

```
function minimumLength(s: string): number {  
  
};
```

C#:

```
public class Solution {  
    public int MinimumLength(string s) {  
  
    }  
}
```

C:

```
int minimumLength(char* s) {  
}  
}
```

Go:

```
func minimumLength(s string) int {  
}  
}
```

Kotlin:

```
class Solution {  
    fun minimumLength(s: String): Int {  
          
    }  
}
```

Swift:

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

Rust:

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

Ruby:

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

PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return Integer  
     */  
    function minimumLength($s) {  
  
    }  
}
```

Dart:

```
class Solution {  
int minimumLength(String s) {  
  
}  
}
```

Scala:

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

Elixir:

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

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int minimumLength(string s) {

    }
};
```

Java Solution:

```
/**
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public int minimumLength(String s) {

    }
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Minimum Length of String After Deleting Similar Ends
Difficulty: Medium
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:

    def minimumLength(self, s: str) -> int:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):

    def minimumLength(self, s):
        """
        :type s: str
        :rtype: int
        """
```

JavaScript Solution:

```
/**
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
```

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

TypeScript Solution:

```
/** 
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function minimumLength(s: string): number {

};
```

C# Solution:

```
/*
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int MinimumLength(string s) {
        }
}
```

C Solution:

```
/*
 * Problem: Minimum Length of String After Deleting Similar Ends
 * Difficulty: Medium
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

int minimumLength(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Length of String After Deleting Similar Ends
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minimumLength(s string) int {

}
```

Kotlin Solution:

```
class Solution {
    fun minimumLength(s: String): Int {
        }
    }
}
```

Swift Solution:

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

```
}
```

```
}
```

Rust Solution:

```
// Problem: Minimum Length of String After Deleting Similar Ends
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn minimum_length(s: String) -> i32 {
        let mut left = 0;
        let mut right = s.len() - 1;

        while left < right && s[left] == s[right] {
            left += 1;
            right -= 1;
        }

        if left >= right {
            return 0;
        } else {
            return right - left + 1;
        }
    }
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

    /**
     * @param String $s
     * @return Integer
     */
    function minimumLength($s) {
        $left = 0;
        $right = strlen($s) - 1;

        while ($left < $right && $s[$left] == $s[$right]) {
            $left++;
            $right--;
        }

        if ($left >= $right) {
            return 0;
        } else {
            return $right - $left + 1;
        }
    }
}
```

Dart Solution:

```
class Solution {  
    int minimumLength(String s) {  
  
    }  
}
```

Scala Solution:

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

Elixir Solution:

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

Erlang Solution:

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

Racket Solution:

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