

Problem 3223: Minimum Length of String After Operations

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a string

s

.

You can perform the following process on

s

any

number of times:

Choose an index

i

in the string such that there is

at least

one character to the left of index

i

that is equal to

$s[i]$

, and

at least

one character to the right that is also equal to

$s[i]$

.

Delete the

closest

occurrence of

$s[i]$

located to the

left

of

i

.

Delete the

closest

occurrence of

`s[i]`

located to the

right

of

i

.

Return the

minimum

length of the final string

s

that you can achieve.

Example 1:

Input:

`s = "abaacbcbb"`

Output:

5

Explanation:

We do the following operations:

Choose index 2, then remove the characters at indices 0 and 3. The resulting string is

`s = "bacbcbb"`

.

Choose index 3, then remove the characters at indices 0 and 5. The resulting string is

`s = "acbccb"`

.

Example 2:

Input:

`s = "aa"`

Output:

2

Explanation:

We cannot perform any operations, so we return the length of the original string.

Constraints:

$1 \leq s.length \leq 2 * 10$

5

s

consists only of lowercase English letters.

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 Operations
 * Difficulty: Medium
 * Tags: string, hash
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

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

    }
};
```

Java Solution:

```
/**
 * Problem: Minimum Length of String After Operations
```

```

* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minimumLength(String s) {

}

}

```

Python3 Solution:

```

"""
Problem: Minimum Length of String After Operations
Difficulty: Medium
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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 Operations
 * Difficulty: Medium
 * Tags: string, hash
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

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

};

```

TypeScript Solution:

```

/**
 * Problem: Minimum Length of String After Operations
 * Difficulty: Medium
 * Tags: string, hash
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

function minimumLength(s: string): number {

};

```

C# Solution:

```

/*
 * Problem: Minimum Length of String After Operations
 * Difficulty: Medium
 * Tags: string, hash
 *
 * Approach: String manipulation with hash map or two pointers

```

```

* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}

}

```

C Solution:

```

/*
* Problem: Minimum Length of String After Operations
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minimumLength(char* s) {

}

```

Go Solution:

```

// Problem: Minimum Length of String After Operations
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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 Operations
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

    }
}

```

Ruby Solution:

```

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

end

```

PHP Solution:

```

class Solution {

```

```

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

}
}

```

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?)
)
```