Problem 3412: Find Mirror Score of a String

Problem Information

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

Problem Description

You are given a string

s

.

We define the

mirror

of a letter in the English alphabet as its corresponding letter when the alphabet is reversed. For example, the mirror of

'a'

is

'z'

, and the mirror of

'y'

is

'b'

.

Initially, all characters in the string

s

are

unmarked

.

You start with a score of 0, and you perform the following process on the string

s

:

Iterate through the string from left to right.

At each index

i

, find the closest

unmarked

index

j

such that

j < i

and

s[j]

is the mirror of

s[i]

. Then,

mark

both indices

i

and

j

, and add the value

i - j

to the total score.

If no such index

j

exists for the index

i

, move on to the next index without making any changes.

Return the total score at the end of the process.

Example 1:

Input:

s = "aczzx"

Output:

5

Explanation:

i = 0

. There is no index

j

that satisfies the conditions, so we skip.

i = 1

. There is no index

j

that satisfies the conditions, so we skip.

i = 2

. The closest index

j

that satisfies the conditions is

j = 0

, so we mark both indices 0 and 2, and then add

2 - 0 = 2

to the score.

i = 3

. There is no index

j

that satisfies the conditions, so we skip.

i = 4

. The closest index

j

that satisfies the conditions is

j = 1

, so we mark both indices 1 and 4, and then add

4 - 1 = 3

to the score.

Example 2:

Input:

s = "abcdef"

Output:

0

Explanation:

For each index

i

, there is no index

j

that satisfies the conditions.

Constraints:

1 <= s.length <= 10

5

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
long long calculateScore(string s) {

}
};
```

Java:

```
class Solution {
public long calculateScore(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long CalculateScore(string s) {

}
}
```

C:

```
long long calculateScore(char* s) {

}
```

Go:

```
func calculateScore(s string) int64 {

}
```

Kotlin:

```
class Solution {
fun calculateScore(s: String): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn calculate_score(s: String) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
```

```
*/
function calculateScore($s) {

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def calculateScore(s: String): Long = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (calculate-score s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Mirror Score of a String
* Difficulty: Medium
* Tags: string, hash, stack
*
* 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:
long long calculateScore(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Find Mirror Score of a String
* Difficulty: Medium
* Tags: string, hash, stack
*
* 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 long calculateScore(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Mirror Score of a String
Difficulty: Medium
Tags: string, hash, stack

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Mirror Score of a String
* Difficulty: Medium
* Tags: string, hash, stack
*
* 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 calculateScore = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Find Mirror Score of a String
* Difficulty: Medium
* Tags: string, hash, stack
*
* 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 calculateScore(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Find Mirror Score of a String
* Difficulty: Medium
* Tags: string, hash, stack
*
* 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 long CalculateScore(string s) {

}
}
```

C Solution:

```
/*
* Problem: Find Mirror Score of a String
* Difficulty: Medium
```

```
* Tags: string, hash, stack
*
* 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
*/

long long calculateScore(char* s) {

}
```

Go Solution:

```
// Problem: Find Mirror Score of a String
// Difficulty: Medium
// Tags: string, hash, stack
//
// 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 calculateScore(s string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun calculateScore(s: String): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Mirror Score of a String
// Difficulty: Medium
// Tags: string, hash, stack
//
// 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 calculate_score(s: String) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

```
object Solution {
def calculateScore(s: String): Long = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (calculate-score s)
(-> string? exact-integer?)
)
```