Problem 3110: Score of a String

Problem Information

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

Problem Description

You are given a string

s

. The

score

of a string is defined as the sum of the absolute difference between the

ASCII

values of adjacent characters.

Return the

score

of

s

.

Example 1:

Input:

s = "hello"

Output:

13

Explanation:

The

ASCII

values of the characters in

s

are:

'h' = 104

,

'e' = 101

,

'l' = 108

,

'o' = 111

. So, the score of

s

would be

|104 - 101| + |101 - 108| + |108 - 108| + |108 - 111| = 3 + 7 + 0 + 3 = 13

.

Example 2:

Input:

s = "zaz"

Output:

50

Explanation:

The

ASCII

values of the characters in

s

are:

'z' = 122

,

'a' = 97

. So, the score of

s

would be

|122 - 97| + |97 - 122| = 25 + 25 = 50

.

Constraints:

2 <= s.length <= 100

s

consists only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def scoreOfString(self, s):
```

```
"""
:type s: str
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int scoreOfString(char* s) {

}
```

Go:

```
func scoreOfString(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int scoreOfString(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Score of a String
Difficulty: Easy
Tags: string
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* 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 scoreOfString = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function scoreOfString(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```
/*
* Problem: Score of a String
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

int scoreOfString(char* s) {

}
```

Go Solution:

```
// Problem: Score of a String
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func scoreOfString(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Score of a String
// Difficulty: Easy
// Tags: string
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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