Problem 2223: Sum of Scores of Built Strings

Problem Information

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

Problem Description

You are

building

a string

s

of length

n

one

character at a time,

prepending

each new character to the

front

of the string. The strings are labeled from

1

to

n

, where the string with length

i

is labeled

s

i

.

For example, for

s = "abaca"

,

s

1

== "a"

,

s

2

== "ca"

,

s

3

== "aca"

, etc.

The

score

of

s

i

is the length of the

longest common prefix

between

s

i

and

s

n

(Note that

s == s

n

).

Given the final string

s

, return

the

sum

of the

score

of every

s

i

.

Example 1:

Input:

s = "babab"

Output:

9

Explanation:

For s

1

== "b", the longest common prefix is "b" which has a score of 1. For s

2

== "ab", there is no common prefix so the score is 0. For s

3

== "bab", the longest common prefix is "bab" which has a score of 3. For s

4

== "abab", there is no common prefix so the score is 0. For s

5

== "babab", the longest common prefix is "babab" which has a score of 5. The sum of the scores is 1 + 0 + 3 + 0 + 5 = 9, so we return 9.

Example 2:

Input:

s = "azbazbzaz"

Output:

14

Explanation:

For s

2

== "az", the longest common prefix is "az" which has a score of 2. For s

6

== "azbzaz", the longest common prefix is "azb" which has a score of 3. For s

9

== "azbazbzaz", the longest common prefix is "azbazbzaz" which has a score of 9. For all other s

i

, the score is 0. The sum of the scores is 2 + 3 + 9 = 14, so we return 14.

Constraints:

1 <= s.length <= 10

5

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

```
func sumScores(s string) int64 {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (sum-scores s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
long long sumScores(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public long sumScores(String s) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Sum of Scores of Built Strings
Difficulty: Hard
Tags: array, string, hash, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function sumScores(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public long SumScores(string s) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Scores of Built Strings
* Difficulty: Hard
* Tags: array, string, hash, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

long long sumScores(char* s) {

}
```

Go Solution:

```
// Problem: Sum of Scores of Built Strings
// Difficulty: Hard
// Tags: array, string, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func sumScores(s string) int64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Sum of Scores of Built Strings
// Difficulty: Hard
// Tags: array, string, hash, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (sum-scores s)
(-> string? exact-integer?)
)
```