Problem 58: Length of Last Word

Problem Information

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

Problem Description

Given a string

s

consisting of words and spaces, return

the length of the

last

word in the string.

A

word

is a maximal

substring

consisting of non-space characters only.

Example 1:

Input:

s = "Hello World"

Output:

5

Explanation:

The last word is "World" with length 5.

Example 2:

Input:

s = " fly me to the moon "

Output:

4

Explanation:

The last word is "moon" with length 4.

Example 3:

Input:

s = "luffy is still joyboy"

Output:

6

Explanation:

The last word is "joyboy" with length 6.

Constraints:

1 <= s.length <= 10

4

s

consists of only English letters and spaces

''

.

There will be at least one word in

s

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int lengthOfLastWord(char* s) {

}
```

Go:

```
func lengthOfLastWord(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (length-of-last-word s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Length of Last Word
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Length of Last Word
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int lengthOfLastWord(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Length of Last Word
Difficulty: Easy
Tags: string, tree

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Length of Last Word
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

```
};
```

TypeScript Solution:

```
/**
* Problem: Length of Last Word
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function lengthOfLastWord(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Length of Last Word
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Length of Last Word
* Difficulty: Easy
```

```
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int lengthOfLastWord(char* s) {

}
```

Go Solution:

```
// Problem: Length of Last Word
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func lengthOfLastWord(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Length of Last Word
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (length-of-last-word s)
(-> string? exact-integer?)
)
```