58. Length of Last Word

Previous Next

Given a string `s` consists of upper/lower-case alphabets and empty space characters `' '`, return the length of last word (last word means the last appearing word if we loop from left to right) in the string.

If the last word does not exist, return 0.

Note: A word is defined as a **maximal substring** consisting of non-space characters only.

Example:

```
Input: "Hello World"
Output: 5
```

Solution

Approach 1: String Index Manipulation

Intuition

There is no doubt that this is an easy problem. Yet, it could be a good exercise for one to practice string manipulation, which is definitely common during interviews.

In this article, we start with some approaches that manipulate string indexes, then we look at how to use the built-in string functions to solve the problem.

> The intuition is simple, as it pretty much given away from the name of the problem, *i.e.* first we **locate** the last word, then we **count** the length of the last word.

One should pay attention to some edge cases though:

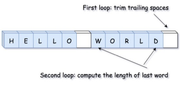
- The input string could be empty.
- There could be some trailing spaces in the input string, *e.g.* `hello <space>`.
- There might only be one word in the given string.

The challenge is to build a concise yet comprehensive solution that could handle all above cases.

Algorithm

One can break down the solution into two steps:

- First, we would try to locate the last word, starting from the end of the string. We iterate the string in reverse order, consuming the empty spaces. When we first come across a non-space character, we know that we are at the last character of the last word.
- Second, once we locate the last word. We count its length, starting from its last character. Again, we could use a loop here.



Here is what it looks like, a solution with **two loops**:

Java Python3 Copy

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        # trim the trailing spaces
        p = len(s) - 1
        while p >= 0 and s[p] == ' ':
            p -= 1

        # compute the length of last word
        length = 0
        while p >= 0 and s[p] != ' ':
            p -= 1
            length += 1
        return length
```

Complexity

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

  In the worst case, the input string might contain only a single word, which implies that we would need to iterate through the entire string to obtain the result.

- Space Complexity: $\mathcal{O}(1)$, only constant memory is consumed, regardless the input.

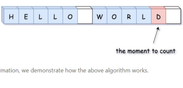
Approach 2: One-loop Iteration

Intuition

In the above approach, we applied two loops. One is used to locate the last word, and the other one to calculate its length.

> We could actually complete the same tasks within a single loop.

The trick is that we could define a condition, *i.e.* the precise moment that we should start to count the length of the word.



In the following animation, we demonstrate how the above algorithm works.

Algorithm

Here are some sample implementations with comments.

Java Python3 Copy

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        p, length = len(s), 0

        while p > 0:
            p -= 1
            # if we are in the middle of the last word
            if s[p] != ' ':
                length += 1
            # here is the end of last word
            elif length > 0:
                return length

        return length
```

Complexity

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

  This approach has the same time complexity as the previous approach. The only difference is that we combined two loops into one.

- Space Complexity: $\mathcal{O}(1)$, again a constant memory is consumed, regardless the input.

Approach 3: Built-in String Functions

Intuition

As we mentioned at the beginning of the article, we could also resort to the ***built-in*** functions of the String data structure, in order to solve the problem.

In fact, String is such an important data type in many programming languages, that often it comes with a rich set of built-in functions that can help one to accomplish many common tasks, such as trimming the empty spaces in the string etc.

It would be really helpful to get proficient with those built-in functions.

Algorithm

In different programming languages, the sets of built-in functions associated with String are different though. In this section, we showcase some examples.

In Python, we would use the following built-in functions to accomplish the tasks:

- `str.isspace()`: this function determines if the `str` contains only spaces.
- `str.split(delimiter)`: this function could split the input string into several substrings, based on the given delimiter (by default, the delimiter is space).

One can find the list of built-in functions in Python from the official documentation.

In Java, here is a list of built-in functions that we would use:

- `String.trim()`: this function returns a copy of the string, with the leading and trailing whitespaces trimmed.
- `String.length()`: this function returns the length of the string.
- `String.lastIndexOf(char)`: this function returns the index of the last occurrence of the given character.

Again, one can find more details about the APIs for the String data structure in Java, from the official documentation.

Java Python3 Copy

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        return 0 if not s or s.isspace() else len(s.split()[-1])
```

Complexity Analysis

- Time Complexity: $\mathcal{O}(N)$, where N is the length of the input string.

  Since we use some built-in function from the String data type, we should look into the complexity of each built-in function that we used, in order to obtain the overall time complexity of our algorithm.

  It would be safe to assume the time complexity of the methods such as `str.split()` and `String.lastIndexOf()` to be $\mathcal{O}(N)$, since in the worst case we would need to scan the entire string for both methods.

- Space Complexity: $\mathcal{O}(N)$. Again, we should look into the built-in functions that we used in the algorithm.

  In the Java implementation, we used the function `String.trim()` which returns a ***copy*** of the input string without leading and trailing whitespace. Therefore, we would need $\mathcal{O}(N)$ space for our algorithm to hold this copy.

  In the Python implementation, we used `str.split()`, which returns a list of ***substrings*** that are separated by the space delimiter. As a result, we would need $\mathcal{O}(N)$ space for our algorithm to store this list.

Rate this article:

Previous Next

Comments: 6

Sort By

Type comment here... (Markdown is supported)

Preview Post

londheimanshu 0 2 days ago
Python3 solution (36 ms) (0%)

```
def lengthOfLastWord(self, s: str) -> int:
    # s = s.strip()
```

Read More

0 Share Reply

user4667k 0 June 24, 2020 9:17 AM Report

One line JS:

```
var lengthOfLastWord = function(s) {
  return s.trim().split(' ').pop().length;
};
```

Read More

0 Share Reply

rajagosk 1 June 25, 2020 8:07 PM

Any Python solution?

0 Share Reply

colleenpurcell 0 June 16, 2020 10:39 AM

JAVASCRIPT solution 60 ms

```
var lengthOfLastWord = function(s) {
    let arr = s.trim().split(' ')
    // console.log(arr[arr.length-1])
```

Read More

0 Share Reply

Hongbin 2 May 22, 2020 6:22 AM Report

do we need to use char at?
class Solution {
public int lengthOfLastWord(String s) {
String[] words = s.split(" ");
// System.out.prin...

Read More

0 Share Reply

LeetCodeUser 0 May 6, 2020 10:34 PM

The problem description is confusing. It says "If the last word does not exist, return 0." So, I had a "return 0" at the end. Turns out that we just need to return len regardless. Anyway, here's my code:

```
class Solution {
    // TODO: Get rid of this
```

Read More

0 Share Reply

nancc 0 April 21, 2020 11:14 PM Report

I was surprised that my test case of "un élève" gave an expected output of 7 instead of 5.

0 Share Reply

SHOW 2 REPLIES

haining1527 7 April 21, 2020 3:40 AM

About the approach one, it's one loop right? lines 5-6 just to find the first non-space char at the end of the list. and if we find one, we keep going to calculate the length of the last word. until we meet the next space char, then we return the length.

0 Share Reply