Length of Last Word

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.

Algorithm:

1.) Input Handling:
- Accept a string s as input, which may contain leading or trailing spaces and words separated by spaces.

2.) Remove Extra Spaces:
- Use the strip() method to remove any leading or trailing whitespace from the string s. This results in a new string string without extra spaces at the ends.

3.) Split the String:
- Use the split(" ") method to split the trimmed string string into a list of words, using a single space as the delimiter. This list is stored in the variable length.

4.) Access the Last Word:
- Retrieve the last word from the list length using length[-1].

5.) Compute the Length:
- Calculate the length of the last word using the len() function.

6.) Return the Result:
- Return the length of the last word as the result.

Pseudocode:

```
Function lengthOfLastWord(s):

  # Step 1: Remove leading and trailing spaces

  trimmedString ← Trim(s)


  # Step 2: Split the string into words

  wordsList ← Split(trimmedString, " ")


  # Step 3: Get the last word

  lastWord ← wordsList[-1]


  # Step 4: Return the length of the last word

  Return Length(lastWord)
```

Code:

```
class Solution(object):
   def lengthOfLastWord(self, s):
      string = s.strip()
      length = string.split(" ")
      return len(length[-1])
```

Conclusion:

The provided code efficiently computes the length of the last word in a given string by first trimming unnecessary spaces, splitting the string into words, and then determining the length of the last word. This approach ensures accurate results even when the input string has irregular spacing. The algorithm is simple, easy to understand, and works well for most practical scenarios, with a time complexity of O(n).