557. Reverse Words in a String III

April 8, 2017 | 73.6K views

Average Rating: 3.02 (46 votes)

Given a string, you need to reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

Example 1:

```
Input: "Let's take LeetCode contest"
Output: "s'teL ekat edoCteeL tsetnoc"
```

Note: In the string, each word is separated by single space and there will not be any extra space in the string.

Solution

Approach #1 Simple Solution[Accepted]

The first method is really simple. We simply split up the given string based on whitespaces and put the individual words in an array of strings. Then, we reverse each individual string and concatenate the result. We return the result after removing the additional whitespaces at the end.

Java

Copy

```
public class Solution {
    public String reverseWords(String s) {
        String words[] = s.split(" ");
        StringBuilder res=new StringBuilder();
        for (String word: words)
            res.append(new StringBuffer(word).reverse().toString() + " ");
        return res.toString().trim();
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. where n is the length of the string.
- Space complexity : $O(n)$. res of size n is used.

Approach #2 Without using pre-defined split and reverse function [Accepted]

Algorithm

We can create our own split and reverse function. Split function splits the string based on the delimiter " " (space) and returns the array of words. Reverse function returns the string after reversing the characters.

Java

Copy

```
public class Solution {
    public String reverseWords(String s) {
        String words[] = split(s);
        StringBuilder res=new StringBuilder();
        for (String word: words)
            res.append(reverse(word) + " ");
        return res.toString().trim();
    }
    public String[] split(String s) {
        ArrayList < String > words = new ArrayList < > ();
        StringBuilder word = new StringBuilder();
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == ' ') {
                words.add(word.toString());
                word = new StringBuilder();
            } else
                word.append( s.charAt(i));
        }
        words.add(word.toString());
        return words.toArray(new String[words.size()]);
    }
    public String reverse(String s) {
      StringBuilder res=new StringBuilder();
        for (int i = 0; i < s.length(); i++)
            res.insert(0,s.charAt(i));
        return res.toString();
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. where n is the length of the string.
- Space complexity : $O(n)$. res of size n is used.

Approach #3 Using StringBuilder and reverse method [Accepted]

Algorithm

Instead of using split method, we can use temporary string $word$ to store the word. We simply append the characters to the $word$ until `' '` character is not found. On getting `' '` we append the reverse of the $word$ to the resultant string $result$. Also after completion of loop , we still have to append the $reverse$ of the $word$(last word) to the $result$ string.

Below code is inspired by @ApolloX.

Java

Copy

```
public class Solution {
    public String reverseWords(String input) {
        final StringBuilder result = new StringBuilder();
        final StringBuilder word = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            if (input.charAt(i) != ' ') {
                word.append(input.charAt(i));
            } else {
                result.append(word.reverse());
                result.append(" ");
                word.setLength(0);
            }
        }
        result.append(word.reverse());
        return result.toString();
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. Single loop upto n is there, where n is the length of the string.
- Space complexity : $O(n)$. $result$ and $word$ size will grow upto n.

Rate this article:

Previous Next

Comments: 63

Sort By

Type comment here... (Markdown is supported)

Preview Post

abhi25 ★66 August 13, 2018 11:43 AM

One line python code

```
return " ".join([i[::-1] for i in s.split()])
```

55 Share Reply

SHOW 3 REPLIES

dumbleeter ★269 May 3, 2019 2:52 AM

Isn't using built-in functions defeating the entire purpose of this exercise?

Will it be acceptable in an interview setting?

24 Share Reply

SHOW 3 REPLIES

vadimtukaev ★119 June 17, 2018 10:22 PM

All of these solutions are over-engineered and inefficient, IMHO. Here is a universal algorithm that can be transferred to any language, including those, that using immutable strings (with a few obvious alterations).

Read More

17 Share Reply

SHOW 2 REPLIES

cwoodrow ★10 July 2, 2018 7:02 PM

Not optimal but quick Java solution :D :

```
class Solution {
    public String reverseWords(String s) {
        return Arrays.stream(s.split(" "))
```

Read More

10 Share Reply

DhruvinPatel ★11 January 24, 2019 7:19 AM

Simple Python Solution: faster than 100% solutions

```
def reverseWords(self, s):
    """
    :type s: str
```

Read More

8 Share Reply

frozenwater ★49 January 10, 2019 1:49 PM

Sharing two pointers approach (Java) with O(n) time, O(n) space.

```
class Solution {
    public String reverseWords(String s) {
        int i = 0, l = 0;
```

Read More

3 Share Reply

Summon528 ★3 February 10, 2019 5:05 PM

Python one liner, faster than 100.00%

```
    def reverseWords(self, s: 'str') -> 'str':
        return " ".join(i[::-1] for i in s.split())
```

3 Share Reply

SHOW 1 REPLY

sadykov ★6 January 20, 2019 8:37 PM

Problem states that we should preserve whitespaces. Provided algorithms will fail to correctly reserver the following sentence:

```
"Let's take LeetCode  contest"
```

1 Share Reply

SHOW 1 REPLY

sjw214 ★183 November 29, 2018 4:59 AM

Javascript solution:

```
String.prototype.reverse = function() {
    return this.split('').reverse().join('');
}
```

Read More

1 Share Reply

artizen ★6 May 8, 2018 8:36 AM

```
/*
No use of standard library or stack
*/
std::string reverseWords(std::string s) {
```

Read More

1 Share Reply