151. Reverse Words in a String (/problems/reverse-words-in-a-string/)

Dec. 1, 2019 | 30.9K views

Average Rating: 4.87 (37 votes)

Given an input string, reverse the string word by word.

Example 1:

```
Input: "the sky is blue"
Output: "blue is sky the"
```

Example 2:

```
Input: "  hello world!  "
Output: "world! hello"
Explanation: Your reversed string should not contain leading or trailing spaces.
```

Example 3:

```
Input: "a good   example"
Output: "example good a"
Explanation: You need to reduce multiple spaces between two words to a single sp
```

Note:

- A word is defined as a sequence of non-space characters.
- Input string may contain leading or trailing spaces. However, your reversed string should not contain leading or trailing spaces.

- You need to reduce multiple spaces between two words to a single space in the reversed string.

Follow up:

For C programmers, try to solve it *in-place* in *O*(1) extra space.

Solution

Overview

Different interviewers would probably expect different approaches for this problem. The holy war question is to use or not use built-in methods. As you may notice, most of design problems on Leetcode are voted down because of two main reasons:

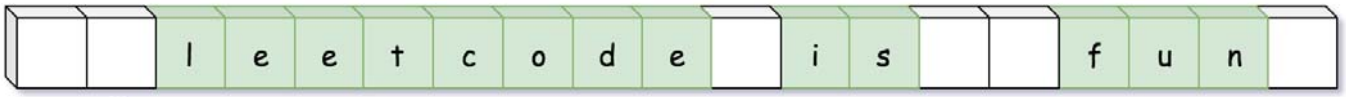
1. There was no approach with built-in methods / data structures in the article.
2. One of the approaches in the article did contain built-in methods / data structures.

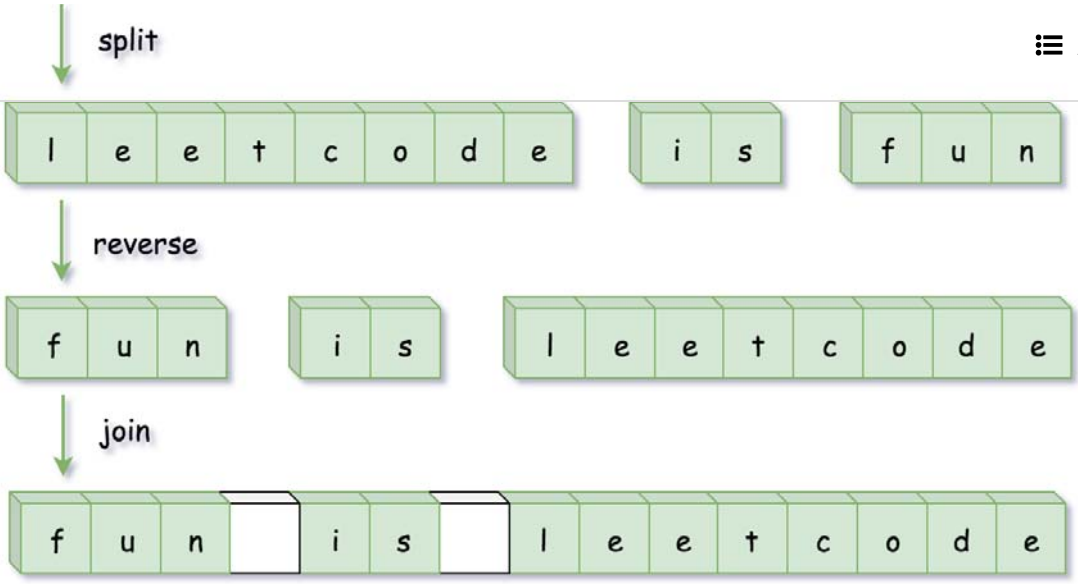
Seems like the community has no common opinion yet, and in practice that means an unpredictable interview experience for some sort of problems.

Here we consider three different solutions of linear time and space complexity:

1. Use built-in split and reverse. Benefits: in-place in Python (in-place, but linear space complexity!) and the simplest one to write.
2. The most straightforward one. Trim the whitespaces, reverse the whole string and then reverse each word.
   Benefits: could be done in-place for the languages with mutable strings.
3. Two passes approach with a deque. Move along the string, word by word, and push each new word in front of the deque. Convert the deque back into string. Benefits: two passes.

Approach 1: Built-in Split + Reverse





Implementation

Java Python Copy

```
class Solution:
    def reverseWords(self, s: str) -> str:
        return " ".join(reversed(s.split()))
```

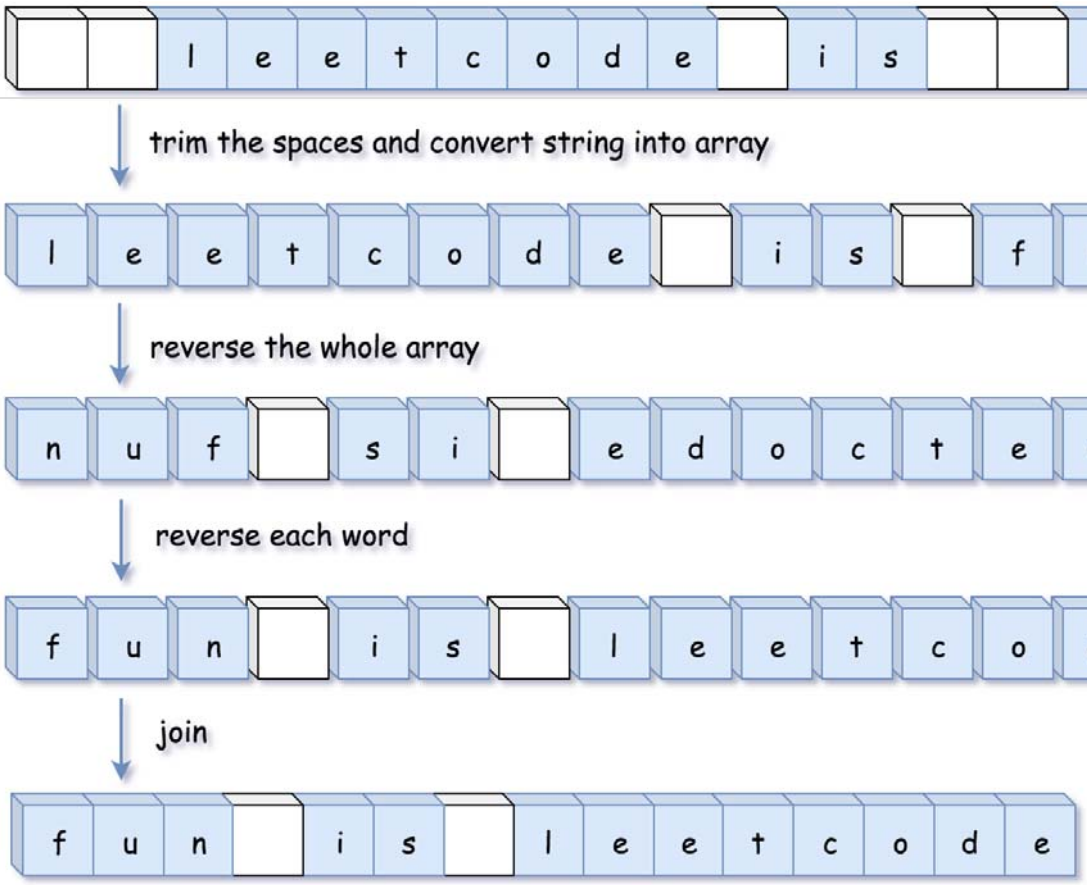
Complexity Analysis

- Time complexity: $\mathcal{O}(N)$, where N is a number of characters in the input string.
- Space complexity: $\mathcal{O}(N)$, to store the result of split by spaces.

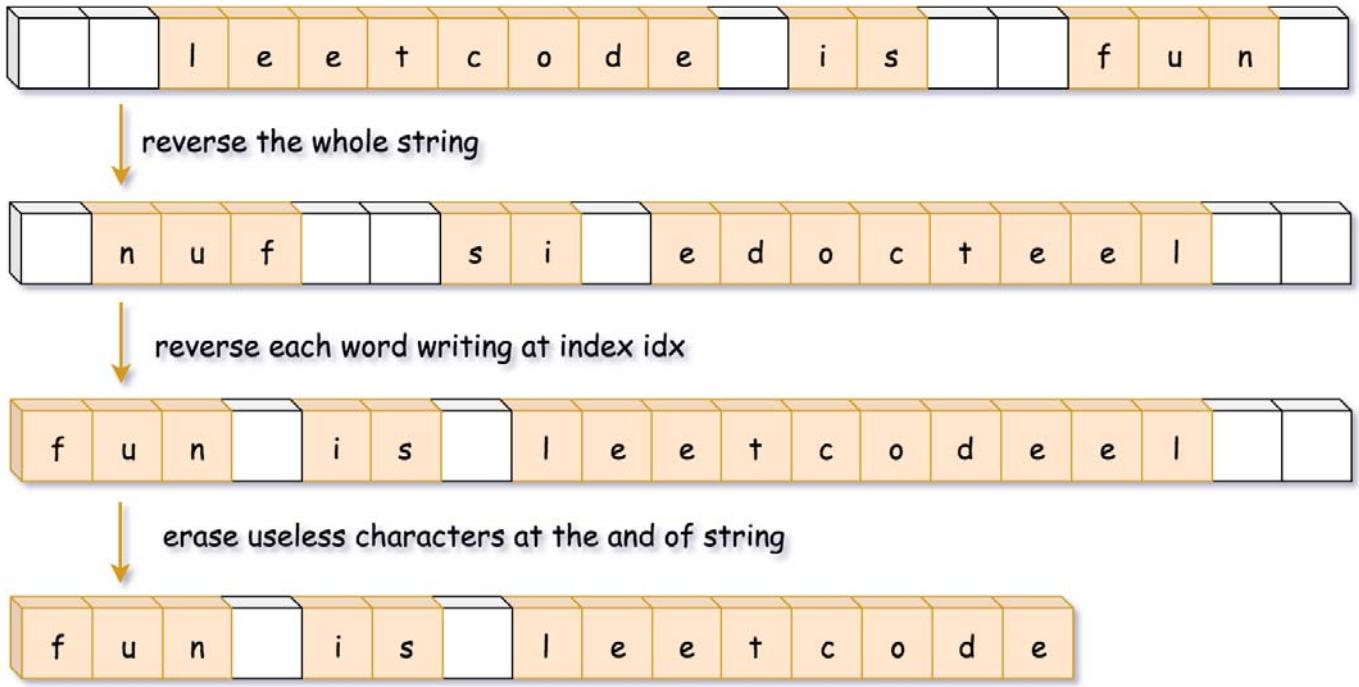
Approach 2: Reverse the Whole String and Then Reverse Each Word

The implementation of this approach will be different for Java/Python (= immutable strings) and C++ (= mutable strings).

In the case of immutable strings one has first to convert string into mutable data structure, and hence it makes sense to trim all spaces during that conversion.



In the case of *mutable* strings, there is no need to allocate an additional data structure, one could make all job done in-place. In such a case it makes sense to reverse words and trim spaces at the same time.



Implementation

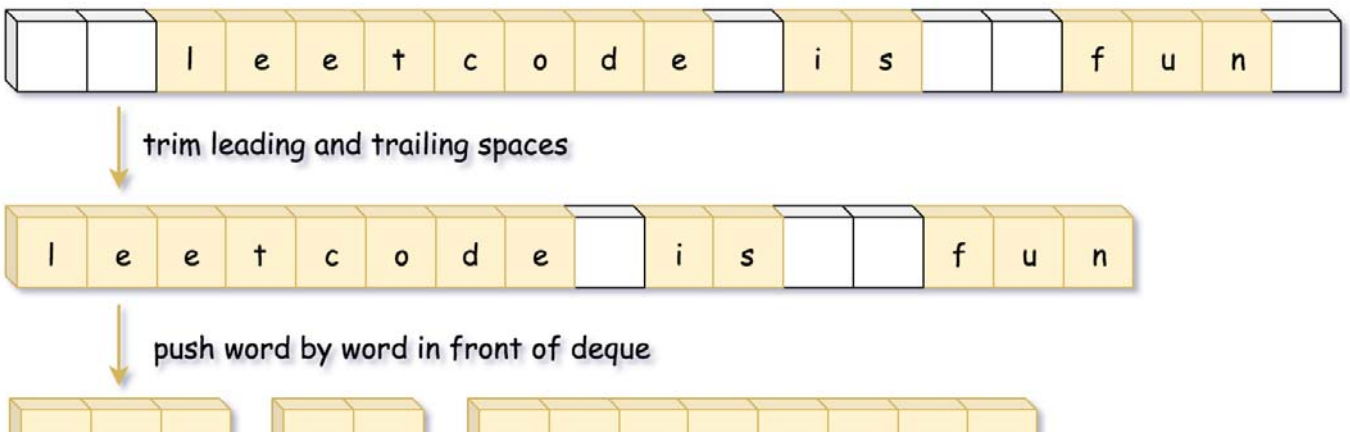
C++ Java Python Copy

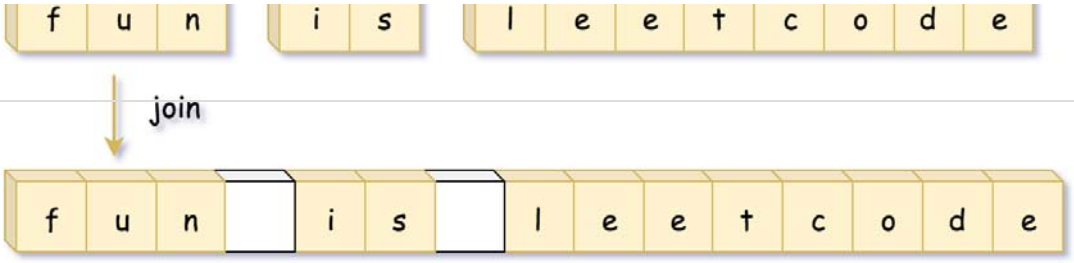
```python
class Solution:
    def trim_spaces(self, s: str) -> list:
        left, right = 0, len(s) - 1
        # remove leading spaces
        while left <= right and s[left] == ' ':
            left += 1

        # remove trailing spaces
        while left <= right and s[right] == ' ':
            right -= 1

        # reduce multiple spaces to single one
        output = []
        while left <= right:
            if s[left] != ' ':
                output.append(s[left])
            elif output[-1] != ' ':
                output.append(s[left])
            left += 1

        return output

    def reverse(self, l: list, left: int, right: int) -> None:
        while left < right:
            l[left], l[right] = l[right], l[left]
            left, right = left + 1, right - 1
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N)$.
- Space complexity: $\mathcal{O}(N)$.

Approach 3: Deque of Words





Implementation

Java | Python

Copy

```python
from collections import deque
class Solution:
    def reverseWords(self, s: str) -> str:
        left, right = 0, len(s) - 1
        # remove leading spaces
        while left <= right and s[left] == ' ':
            left += 1

        # remove trailing spaces
        while left <= right and s[right] == ' ':
            right -= 1

        d, word = deque(), []
        # push word by word in front of deque
        while left <= right:
            if s[left] == ' ' and word:
                d.appendleft(''.join(word))
                word = []
            elif s[left] != ' ':
                word.append(s[left])
            left += 1
        d.appendleft(''.join(word))

        return ' '.join(d)
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N)$.
- Space complexity: $\mathcal{O}(N)$.

Rate this article:

Previous (/articles/flatten-a-multilevel-doubly-linked-list/) Next (/articles/two-sum-iii-data-structure-design/)

Comments: 17

Sort By

Type comment here... (Markdown is supported)

Preview Post

(/rbpal)

rbpal (/rbpal) ★ 39 December 4, 2019 2:00 PM

Great explanation! One question. How did you create those awesome diagrams? What tool did you use?

39 Share Reply

SHOW 1 REPLY

(/ailionx)

ailionx (/ailionx) ★ 30 February 8, 2020 4:15 PM

How is this a medium problem?

12 Share Reply

SHOW 2 REPLIES

(/jessc)

JessC (/jessc) ★ 11 December 2, 2019 8:18 PM

five stars for the vivid diagram and explanation, excellent job!

5 Share Reply

(/beingbmc12)

beingbmc12 (/beingbmc12) ★ 472 13 hours ago

You could also just loop through the words in reverse

```
class Solution {
    public String reverseWords(String s) {
        String[] words = s.trim().split(" ");
```

Read More

3 Share Reply

(/biscottigelato)

biscottigelato (/biscottigelato) ★ 9 February 1, 2020 3:03 AM

How is removing extra spaces in between words not cause the time complexity to go up? For regular arrays removing an intermediate space/character would cause the entire remainder of the array to shift? Isn't the ability to remove a character in O(1) the whole point of a Linked List? How is removing up to N single characters in a mutable array an O(N) operation instead of an O(N^2) operation?

3 Share Reply

SHOW 3 REPLIES


(/tikibu)

tikibu (/tikibu) ★ 3 January 9, 2020 1:57 AM

Great analysis, thank you! For some reason, I missed the obvious with two reverses.
(Reversing string, then words)

There's also another one ...

Read More

2 Share Reply

SHOW 1 REPLY


(/riseandfallofme)

riseAndFallOfMe (/riseandfallofme) ★ 4 December 5, 2019 2:38 AM

Nice article. Would like to know about the diagram tool you used?

1 Share Reply


(/vijaik)

vijaik (/vijaik) ★ 1 December 3, 2019 3:18 AM

Nice explanation.

1 Share Reply


(/shmidt)

shmidt (/shmidt) ★ 0 May 31, 2020 3:08 PM

This should be an easy question.

0 Share Reply


(/jenniferwcho)

jenniferwcho (/jenniferwcho) ★ 0 8 hours ago

you can also use a stack

Read More

0 Share Reply