344. Reverse String

Oct. 13, 2019 | 267.6K views

Average Rating: 4.76 (72 votes)

Write a function that reverses a string. The input string is given as an array of characters `char[]`.

Do not allocate extra space for another array, you must do this by **modifying the input array** **in-place** with O(1) extra memory.

You may assume all the characters consist of printable ascii characters.

Example 1:

```
Input: ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
```

Example 2:

```
Input: ["H","a","n","n","a","h"]
Output: ["h","a","n","n","a","H"]
```

Solution

Overview

Life is short, use Python. (c)

Python

```
class Solution:
    def reverseString(self, s):
        s.reverse()
```

Speaking seriously, let's use this problem to discuss two things:

- Does *in-place* mean constant space complexity?
- Two pointers approach.

Approach 1: Recursion, In-Place, $\mathcal{O}(N)$ Space

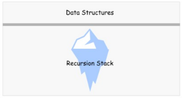
Does *in-place* mean constant space complexity?

No. By definition, an in-place algorithm is an algorithm which transforms input using no auxiliary data structure.

The tricky part is that space is used by many actors, not only by data structures. The classical example is to use recursive function without any auxiliary data structures.

Is it in-place? Yes.

Is it constant space? No, because of recursion stack.



Algorithm

Here is an example. Let's implement recursive function `helper` which receives two pointers, left and right, as arguments.

- Base case: if `left >= right`, do nothing.
- Otherwise, swap `s[left]` and `s[right]` and call `helper(left + 1, right - 1)`.

To solve the problem, call helper function passing the head and tail indexes as arguments: `return helper(0, len(s) - 1)`.

Implementation

Java Python

```
class Solution:
    def reverseString(self, s):
        def helper(left, right):
            if left < right:
                s[left], s[right] = s[right], s[left]
                helper(left + 1, right - 1)

        helper(0, len(s) - 1)
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ time to perform $N/2$ swaps.
- Space complexity : $\mathcal{O}(N)$ to keep the recursion stack.

Approach 2: Two Pointers, Iteration, $\mathcal{O}(1)$ Space

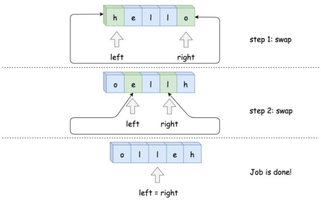
Two Pointers Approach

In this approach, two pointers are used to process two array elements at the same time. Usual implementation is to set one pointer in the beginning and one at the end and then to move them until they both meet.

Sometimes one needs to generalize this approach in order to use three pointers, like for classical Sort Colors problem.

Algorithm

- Set pointer left at index 0, and pointer right at index `n - 1`, where n is a number of elements in the array.
- While left < right:
  - Swap `s[left]` and `s[right]`.
  - Move left pointer one step right, and right pointer one step left.



Implementation

Java Python

```
class Solution:
    def reverseString(self, s):
        left, right = 0, len(s) - 1
        while left < right:
            s[left], s[right] = s[right], s[left]
            left, right = left + 1, right - 1
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ to swap $N/2$ element.
- Space complexity : $\mathcal{O}(1)$, it's a constant space solution.

Rate this article:

Previous Next

Comments: 60

Sort By

Type comment here... (Markdown is supported)

Preview Post

hyankov ★ 259 November 18, 2019 6:19 AM Report

Huh?! Why do we need two pointers again?

```
for (var i = 0; i < s.Length / 2; i++) {
    var tmp = s[i];
```

Read More

112 Share Reply

SHOW 10 REPLIES

terrible_whiteboard ★ 633 May 19, 2020 6:14 PM

I made a video if anyone is having trouble understanding the solution (clickable link)
https://youtu.be/uR6ZZyMQbI

Read More

22 Share Reply

thepatriot ★ 275 November 14, 2019 2:37 AM

excellent article! short, sweet and clear. :)

16 Share Reply

lord909 ★ 10 December 27, 2019 1:13 AM

Why s[i:-1] is not working ?

9 Share Reply

SHOW 6 REPLIES

osmankultur3 ★ 16 January 3, 2020 6:23 PM

OMG :D I literally had a lol moment when I saw the overview: Life is short, use Python. (c)

11 Share Reply

josemb125 ★ 7 March 22, 2020 10:13 PM

hahaha life is short, use python to avoid covid-19

6 Share Reply

heisenberg_blue ★ 9 January 17, 2020 9:04 PM

Can someone explain how the time complexity is O(N) even though we only loop n/2 times? Thanks.

5 Share Reply

SHOW 6 REPLIES

c_jain ★ 28 December 31, 2019 1:48 AM Report

This is the first problem for which I can say C++ one liner.:)

```
for(int i=0; i<s.size()/2; i++) swap(s[i],s[s.size()-i-1]);
// or
```

Read More

5 Share Reply

SHOW 1 REPLY

andot ★ 2 February 15, 2020 11:04 AM

I get it. but the error given for submitting a non-O(1) answer should say so. Instead it was telling me the output was wrong. and I wasted an hour quadruple-checking that wasn't really the case.

2 Share Reply

nilesh3105 ★ 1 June 7, 2020 5:53 PM

1st solution has a tail recursion and will be optimized by the compiler itself

1 Share Reply

SHOW 1 REPLY