186. Reverse Words in a String II

Jan. 19, 2020 | 9.6K views

Average Rating: 4.75 (16 votes)

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

Example:

```
Input:  ["t","h","e"," ","s","k","y"," ","i","s"," ","b","l","u","e"]
Output: ["b","l","u","e"," ","i","s"," ","s","k","y"," ","t","h","e"]
```

Note:

- A word is defined as a sequence of non-space characters.
- The input string does not contain leading or trailing spaces.
- The words are always separated by a single space.

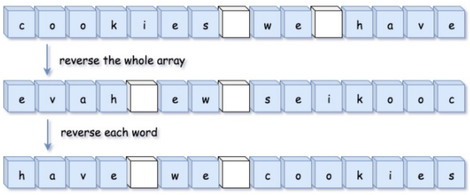
Follow up: Could you do it *in-place* without allocating extra space?

Solution

Approach 1: Reverse the Whole String and Then Reverse Each Word

To have this problem in Amazon interview is a good situation, since input is a mutable structure and hence one could aim $\mathcal{O}(1)$ space solution without any technical difficulties.

> The idea is simple: reverse the whole string and then reverse each word.



Algorithm

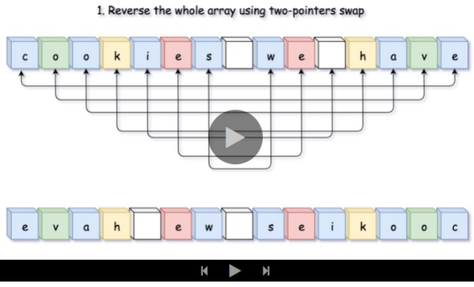
Let's first implement two functions:

- `reverse(l: list, left: int, right: int)`, which reverses array characters between left and right pointers. C++ users could directly use built-in `std::reverse`.
- `reverse_each_word(l: list)`, which uses two pointers to mark the boundaries of each word and previous function to reverse it.

Now `reverseWords(s: List[str])` implementation is straightforward:

- Reverse the whole string: `reverse(s, 0, len(s) - 1)`.
- Reverse each word: `reverse_each_word(s)`.

Implementation



C++ Java Python Copy

```
class Solution {
  public:
  void reverseWords(vector<char>& s) {
    // reverse the whole string
    reverse(s.begin(), s.end());

    int n = s.size();
    int idx = 0;
    for (int start = 0; start < n; ++start) {
      if (s[start] != ' ') {
        // go to the beginning of the word
        if (idx != 0) s[idx++] = ' ';

        // go to the end of the word
        int end = start;
        while (end < n && s[end] != ' ') s[idx++] = s[end++];

        // reverse the word
        reverse(s.begin() + idx - (end - start), s.begin() + idx);

        // move to the next word
        start = end;
      }
    }
  }
};
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N)$, it's two passes along the string.
- Space complexity: $\mathcal{O}(1)$, it's a constant space solution.

Analysis written by @liaison and @andvary

Rate this article:

Previous Next

Comments: 3

Sort By

Type comment here... (Markdown is supported)

Preview Post

lishichengyan ★ 197 April 26, 2020 1:43 AM

clear python code:

```
    def reverseWords(self, s: List[str]) -> None:
        s.reverse()
        tot = len(s)
```

Read More

1 Share Reply

SHOW 1 REPLY

sorianog ★ 8 January 20, 2020 6:34 AM

What is the most optimal solution for this problem with the best complexity analysis results?

1 Share Reply

SHOW 1 REPLY

cihandokur ★ 0 January 27, 2020 2:55 PM

another way

```
public String reverseWords(String text){
        if(text == null || text.length() == 0){
```

Read More

0 Share Reply