> 综合考察字符串操作的好题。

151.翻转字符串里的单词

https://leetcode-cn.com/problems/reverse-words-in-a-string/

给定一个字符串，逐个翻转字符串中的每个单词。

示例 1：
输入: "the sky is blue"
输出: "blue is sky the"

示例 2：
输入: " hello world! "
输出: "world! hello"
解释: 输入字符串可以在前面或者后面包含多余的空格，但是反转后的字符不能包括。

示例 3：
输入: "a good   example"
输出: "example good a"
解释: 如果两个单词间有多余的空格，将反转后单词间的空格减少到只含一个。

思路

这道题目可以说是综合考察了字符串的多种操作。

一些同学会使用split库函数，分隔单词，然后定义一个新的string字符串，最后再把单词倒序相加，那么这道题题目就是一道水题了，失去了它的意义。

所以这里我还是提高一下本题的难度：**不要使用辅助空间，空间复杂度要求为O(1)。**

不能使用辅助空间之后，那么只能在原字符串上下功夫了。

想一下，我们将整个字符串都反转过来，那么单词的顺序指定是倒序了，只不过单词本身也倒叙了，那么再把单词反转一下，单词不就正过来了。

所以解题思路如下：

- 移除多余空格
- 将整个字符串反转
- 将每个单词反转

举个例子，源字符串为："the sky is blue "

- 移除多余空格 : "the sky is blue"
- 字符串反转："eulb si yks eht"
- 单词反转："blue is sky the"

这样我们就完成了翻转字符串里的单词。

思路很明确了，我们说一说代码的实现细节，就拿移除多余空格来说，一些同学会上来写如下代码：

std::string::erase
(2) character: iterator erase (const_iterator p);
(2) character: Erases the character pointed by p.

```
void removeExtraSpaces(string& s) {
    for (int i = s.size() - 1; i > 0; i--) {
        if (s[i] == s[i - 1] && s[i] == ' ') {
            s.erase(s.begin() + i);
        }
    }
    // 删除字符串最后面的空格
    if (s.size() > 0 && s[s.size() - 1] == ' ') {
        s.erase(s.begin() + s.size() - 1);
    }
    // 删除字符串最前面的空格
    if (s.size() > 0 && s[0] == ' ') {
        s.erase(s.begin());
    }
}
```

逻辑很简单，从前向后遍历，遇到空格了就erase。

如果不仔细琢磨一下erase的时间复杂读，还以为以上的代码是O(n)的时间复杂度呢。

想一下真正的时间复杂度是多少，一个erase本来就是O(n)的操作，erase实现原理题目：数组：就移除个元素很难么？，最优的算法来移除元素也要O(n)。

erase操作上面还套了一个for循环，那么以上代码移除冗余空格的代码时间复杂度为O(n^2)。

那么使用双指针法来去移除空格，最后resize（重新设置）一下字符串的大小，就可以做到O(n)的时间复杂度。

如果对这个操作比较生疏了，可以再看一下这篇文章：数组：就移除个元素很难么？是如何移除元素的。

那么使用双指针来移除冗余空格代码如下： fastIndex走的快，slowIndex走的慢，最后slowIndex就标记着移除多余空格后新字符串的长度。

```
void removeExtraSpaces(string& s) {
    int slowIndex = 0, fastIndex = 0; // 定义快指针，慢指针
    // 去掉字符串前面的空格
    while (s.size() > 0 && fastIndex < s.size() && s[fastIndex] == ' ') {
        fastIndex++;
    }
    for (; fastIndex < s.size(); fastIndex++) {
        // 去掉字符串中间部分的冗余空格
        if (fastIndex - 1 > 0
                && s[fastIndex - 1] == s[fastIndex]
                && s[fastIndex] == ' ') {
            continue;
        } else {
            s[slowIndex++] = s[fastIndex];
        }
    }
    if (slowIndex - 1 > 0 && s[slowIndex - 1] == ' ') { // 去掉字符串末尾的空格
        s.resize(slowIndex - 1);
    } else {
        s.resize(slowIndex); // 重新设置字符串大小
    }
}
```

记录字符串前面的空格，然后被覆盖掉

执行continue，跳过多余的空格，fastIndex加1，slowIndex不变；

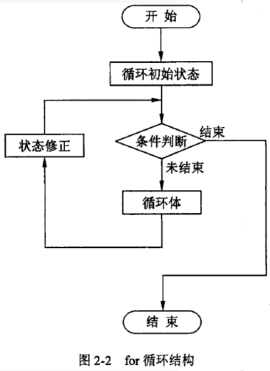


图 2-2 for 循环结构

有的同学可能发现用erase来移除空格，在leetcode上性能也还行。主要是以下几点；：

1. leetcode上的测试集里，字符串的长度不够长，如果足够长，性能差距会非常明显。
2. leetcode的测程序耗时不是很准确的。

此时我们已经实现了removeExtraSpaces函数来移除冗余空格。

还做实现反转字符串的功能，支持反转字符串子区间，这个实现我们分别在344.反转字符串和541.反转字符串II里已经讲过了。

std::string::resize

```
void resize (size_t n);
void resize (size_t n, char c);
```

Resize string

Resizes the string to a length of *n* characters.

If *n* is smaller than the current string length, the current value is shortened to its first *n* character, removing the characters beyond the *n*th.

If *n* is greater than the current string length, the current content is extended by inserting at the end as many characters as needed to reach a size of *n*. If *c* is specified, the new elements are initialized as copies of *c*, otherwise, they are *value-initialized characters* (null characters).

代码如下：

```
// 反转字符串s中左闭又闭的区间[start, end]
void reverse(string& s, int start, int end) {
    for (int i = start, j = end; i < j; i++, j--) {
        swap(s[i], s[j]);
    }
}
```

本题C++整体代码

```
// 版本一
class Solution {
public:
    // 反转字符串s中左闭又闭的区间[start, end]
    void reverse(string& s, int start, int end) {
        for (int i = start, j = end; i < j; i++, j--) {
            swap(s[i], s[j]);
        }
    }

    // 移除冗余空格：使用双指针（快慢指针法）O(n)的算法
    void removeExtraSpaces(string& s) {
        int slowIndex = 0, fastIndex = 0; // 定义快指针，慢指针
        // 去掉字符串前面的空格
        while (s.size() > 0 && fastIndex < s.size() && s[fastIndex] == ' ') {
            fastIndex++;
        }
        for (; fastIndex < s.size(); fastIndex++) {
            // 去掉字符串中间部分的冗余空格
            if (fastIndex - 1 > 0
                    && s[fastIndex - 1] == s[fastIndex]
                    && s[fastIndex] == ' ') {
                continue;
            } else {
                s[slowIndex++] = s[fastIndex];
            }
        }
        if (slowIndex - 1 > 0 && s[slowIndex - 1] == ' ') { // 去掉字符串末尾的空格
            s.resize(slowIndex - 1);
        } else {
            s.resize(slowIndex); // 重新设置字符串大小
        }
    }

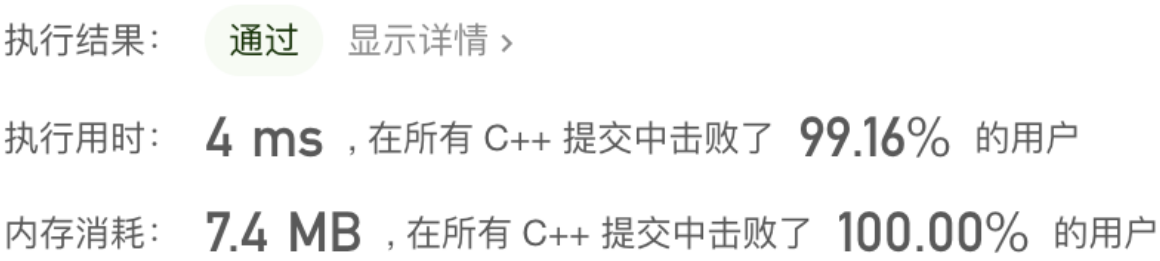
    string reverseWords(string s) {
        removeExtraSpaces(s); // 去掉冗余空格
        reverse(s, 0, s.size() - 1); // 将字符串全部反转
        int start = 0; // 反转的单词在字符串里起始位置
        int end = 0; // 反转的单词在字符串里终止位置
        bool entry = false; // 标记枚举字符串的过程中是否已经进入了单词区间
        for (int i = 0; i < s.size(); i++) { // 开始反转单词
            if (!entry) {
                start = i; // 确定单词起始位置
                entry = true; // 进入单词区间
            }
            // 单词后面有空格的情况，空格就是分词符
            if (entry && s[i] == ' ' && s[i - 1] != ' ') {
                end = i - 1; // 确定单词终止位置
                entry = false; // 结束单词区间
                reverse(s, start, end);
            }
            // 最后一个结尾单词之后没有空格的情况
            if (entry && (i == (s.size() - 1)) && s[i] != ' ' ) {
                end = i;// 确定单词终止位置
                entry = false; // 结束单词区间
                reverse(s, start, end);
```

```
            }
        }
        return s;
    }

    // 当然这里的主函数reverseWords写的有一些冗余的，可以精简一些，精简之后的主函数为：
    /* 主函数简单写法
    string reverseWords(string s) {
        removeExtraSpaces(s);
        reverse(s, 0, s.size() - 1);
        for(int i = 0; i < s.size(); i++) {
            int j = i;
            // 查找单词间的空格，翻转单词
            while(j < s.size() && s[j] != ' ') j++;
            reverse(s, i, j - 1);
            i = j;
        }
        return s;
    }
    */
};
```

双指针

j指向空格，后面赋值给i,循环体结束，然后状态修正i++；



效率：

其他语言版本

Java：

```
class Solution {
   /**
     * 不使用Java内置方法实现
     * <p>
     * 1.去除首尾以及中间多余空格
     * 2.反转整个字符串
     * 3.反转各个单词
     */
    public String reverseWords(String s) {
        // System.out.println("ReverseWords.reverseWords2() called with: s = [" + s + "]");
        // 1.去除首尾以及中间多余空格
        StringBuilder sb = removeSpace(s);
        // 2.反转整个字符串
        reverseString(sb, 0, sb.length() - 1);
        // 3.反转各个单词
        reverseEachWord(sb);
        return sb.toString();
    }

    private StringBuilder removeSpace(String s) {
        // System.out.println("ReverseWords.removeSpace() called with: s = [" + s + "]");
```

```
        int start = 0;
        int end = s.length() - 1;
        while (s.charAt(start) == ' ') start++;
        while (s.charAt(end) == ' ') end--;
        StringBuilder sb = new StringBuilder();
        while (start <= end) {
            char c = s.charAt(start);
            if (c != ' ' || sb.charAt(sb.length() - 1) != ' ') {
                sb.append(c);
            }
            start++;
        }
        // System.out.println("ReverseWords.removeSpace returned: sb = [" + sb + "]");
        return sb;
    }

    /**
     * 反转字符串指定区间[start, end]的字符
     */
    public void reverseString(StringBuilder sb, int start, int end) {
        // System.out.println("ReverseWords.reverseString() called with: sb = [" + sb + "],
        while (start < end) {
            char temp = sb.charAt(start);
            sb.setCharAt(start, sb.charAt(end));
            sb.setCharAt(end, temp);
            start++;
            end--;
        }
        // System.out.println("ReverseWords.reverseString returned: sb = [" + sb + "]");
    }

    private void reverseEachWord(StringBuilder sb) {
        int start = 0;
        int end = 1;
        int n = sb.length();
        while (start < n) {
            while (end < n && sb.charAt(end) != ' ') {
                end++;
            }
            reverseString(sb, start, end - 1);
            start = end + 1;
            end = start + 1;
        }
    }
}
```

```
class Solution:
  #1.去除多余的空格
        def trim_spaces(self,s):
            n=len(s)
            left=0
            right=n-1

            while left<=right and s[left]==' ':       #去除开头的空格
                left+=1
            while left<=right and s[right]==' ':        #去除结尾的空格
```

```
                right=right-1
            tmp=[]
            while left<=right:                                  #去除单词中间多余的空格
                if s[left]!=' ':
                    tmp.append(s[left])
                elif tmp[-1]!=' ':                                  #当前位置是空格，但是相邻
                    tmp.append(s[left])
                left+=1
            return tmp
#2.翻转字符数组
        def reverse_string(self,nums,left,right):
            while left<right:
                nums[left], nums[right]=nums[right],nums[left]
                left+=1
                right-=1
            return None
#3.翻转每个单词
        def reverse_each_word(self, nums):
            start=0
            end=0
            n=len(nums)
            while start<n:
                while end<n and nums[end]!=' ':
                    end+=1
                self.reverse_string(nums,start,end-1)
                start=end+1
                end+=1
            return None

#4.翻转字符串里的单词
        def reverseWords(self, s): #测试用例："the sky is blue"
            l = self.trim_spaces(s)                     #输出：['t', 'h', 'e', ' ', 's', 'k
            self.reverse_string( l,  0, len(l) - 1)   #输出：['e', 'u', 'l', 'b', ' ', 's',
            self.reverse_each_word(l)                #输出：['b', 'l', 'u', 'e', ' ', 'i', '
            return ''.join(l)                                   #输出：blue is sky the


'''


Go：

```go
import (
        "fmt"
)

func reverseWords(s string) string {
        //1.使用双指针删除冗余的空格
        slowIndex, fastIndex := 0, 0
        b := []byte(s)
        //删除头部冗余空格
        for len(b) > 0 && fastIndex < len(b) && b[fastIndex] == ' ' {
                fastIndex++
        }
    //删除单词间冗余空格
        for ; fastIndex < len(b); fastIndex++ {
                if fastIndex-1 > 0 && b[fastIndex-1] == b[fastIndex] && b[fastIndex] == ' '
                        continue
```
```

```
                }
                b[slowIndex] = b[fastIndex]
                slowIndex++
        }
        //删除尾部冗余空格
        if slowIndex-1 > 0 && b[slowIndex-1] == ' ' {
                b = b[:slowIndex-1]
        } else {
                b = b[:slowIndex]
        }
        //2.反转整个字符串
        reverse(&b, 0, len(b)-1)
        //3.反转单个单词  i单词开始位置，j单词结束位置
        i := 0
        for i < len(b) {
                j := i
                for ; j < len(b) && b[j] != ' '; j++ {
                }
                reverse(&b, i, j-1)
                i = j
                i++
        }
        return string(b)
}

func reverse(b *[]byte, left, right int) {
        for left < right {
                (*b)[left], (*b)[right] = (*b)[right], (*b)[left]
                left++
                right--
        }
}
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录