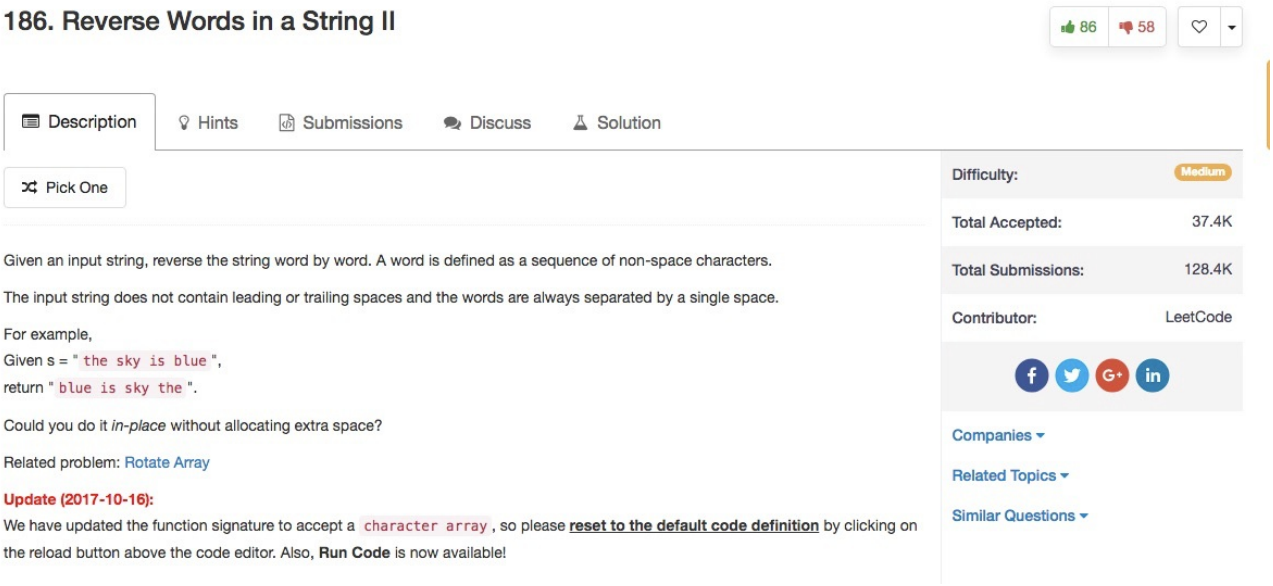
#LeetCode186

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

Yifeng Zeng

#题目描述

Reverse Words in a String II



#思路报告

最开始我是做的Lintcode的53题http://www.lintcode.com/en/problem/reverse-words-in-a-string/ 输入输出都是String而不是char[]，当时的思路是用String[] strs = s.split(" ");把输入String分成多个word。因为String在Java里面本来就是immutable的，所以这需要extra O(n)的space，然后输出也是String，所以可以直接用一个StringBuilder把Sting[] strs，从后往前拼接起来中间加空格即可，然后返回StringBuilder.toString()。所以time O(n)，space O(n)。

代码如下：

```
public String reverseWords(String str) {
        char[] s str.toCharArray();
```

```
        if (s == null || s.length() <= 1) {
            return s;
        }

        String[] strs = s.split(" ");
        StringBuilder sb = new StringBuilder();
        for (int i = strs.length - 1; i >= 0; i--) {
            sb.append(strs[i]);
            if (i != 0) {
                sb.append(" ");
            }
        }

        return sb.toString();
    }
```

LC186是输入char[] s返回为void，所以可以inplace的做space O(1)。由于做了上面lintcode的题我第一反应还是把char[]变成String然后根据上面的步骤得到return的String再String.toCharArray(),但是这样做明显复杂了。时间上都是O(n)，且要遍历整个char[]，所以没法优化了，所有只有优化空间想办法做O(1) space。每个word的长度显然不可能一样，那么左边一个word直接跟右边一个word交换明显太复杂不合适。那么先想到先整体reverse，这样的话每个word的位置已经正确，但是word本身就反过来了，那么再扫一遍把每一个word reverse一遍就可以了。怎么判断每一个word呢？那肯定需要两个指针，i指向word第一个char，j指向word最后一个char，那么j指向一个空格的时候j-1就指向了当前word的最后一个char，再把这个word reverse就好，那么最后一个word后面没有空格，所以扫面完后最后的i，j-1还需要再reverse一下。

代码如下

```
public String reverseWords(String str) {
        char[] s = str.toCharArray();

        if (s == null || s.length <= 1) {
            return new String(s);
        }

        reverse(s, 0, s.length - 1);
        int i = 0;
        int j = 0;
        while (j < s.length) {
            if (s[j] == ' ') {
                reverse(s, i, j-1);
                i = j+1;
            }
            j++;
```

```
        }
        reverse(s, i, j-1);

        return new String(s);
    }

    private void reverse(char[] s, int i, int j) {
        while (i < j) {
            char t = s[i];
            s[i] = s[j];
            s[j] = t;
            i++;
            j--;
        }
    }
```

#套路总结

- 程序的模块化很重要，单独把reverse函数提出来就不用在主函数里面重复写多次了。
- 当时间复杂度不再能够被优化的时候，保持相同时间复杂度的情况下可以考虑优化空间。当然，经常也有通过空间换取时间的情况。

#Follow Up

- 有leading trailing space, 或者每个word之间不止一个space怎么处理。
  - 我觉得应该跟面试官讨论这种情况，看他想要怎么处理，是我的话我应该会说，ok, let's remove the leading trailing spaces for now，先不考虑它，word之间的spaces的话我reverse过来可不可以只留一个space，那么这样的话我就可以用我的第一种方法解了。