# 题目

给定一个字符串s，你需要反转字符串中每个单词的字符顺序，同时仍保留空格和单词的初始顺序。

示例 1：

输入：s = "Let's take LeetCode contest"

输出："s'teL ekat edoCteeL tsetnoc"

示例 2:

输入： s = "Mr Ding"

输出："rM gniD"

提示：

1 <= s.length <= 5 \* 104

s包含可打印的ASCII字符。

s不包含任何开头或结尾空格。

s里至少有一个词。

s中的所有单词都用一个空格隔开。

# 分析

## 方法一：双指针

思路：

你可以按以下步骤实现：

1、遍历字符串，找到每个单词的起始位置和结束位置。

2、对于每个单词，使用双指针技巧，反转单词中的字符。

3、最后将处理后的单词拼接起来。

以下是具体的实现：

class Solution {

public:

string reverseWords(string s) {

int n = s.length();

int start = 0, end = 0;

while (end < n) {

while (end < n && s[end] != ' ') {

end++;

}

reverse(s.begin() + start, s.begin() + end);

start = end + 1;

end++;

}

return s;

}

};

这段代码中，我们首先找到每个单词的起始位置和结束位置，然后使用 reverse 函数反转单词中的字符。最后返回整个字符串。

reverse 函数是标准库 <algorithm> 中的函数，用于反转指定范围内的元素。它接受两个迭代器作为参数，表示要反转的范围的起始和结束位置（不包括结束位置）。例如，reverse(v.begin(), v.end()) 可以将向量 v 中的所有元素反转。

或：

代码：

class Solution {

public:

string reverseWords(string s) {

int length = s.length();

int i = 0;

while (i < length) {

// 找到当前单词的起始位置

int start = i;

// 找到当前单词的结束位置

while (i < length && s[i] != ' ') {

i++;

}

// 反转当前单词

int left = start, right = i - 1;

while (left < right) {

swap(s[left], s[right]);

left++;

right--;

}

// 跳过空格，指向下一个单词的起始位置

while (i < length && s[i] == ' ') {

i++;

}

}

return s;

}

};

复杂度分析

时间复杂度：O(N)。字符串中的每个字符要么在O(1)的时间内被交换到相应的位置，要么因为是空格而保持不动。

空间复杂度：O(1)。因为不需要开辟额外的数组。