# 557. Reverse Words in a String III

---

## Question :

Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and the initial word order.

---

## Constraints :

- $1 \leq s.length \leq 5 \times 10^4$
- s contains printable ASCII characters.
- s does not contain any leading or trailing spaces.
- There is at least one word in s.
- All the words in sss are separated by a single space.

---

## Inputs :

- s: A string containing words separated by single spaces.

## Outputs :

- The string sss, with each word reversed while preserving whitespace and word order.

---

## Example 1 :

Input:
s = "Let's take LeetCode contest"

Output:
"s'teL ekat edoCteeL tsetnoc"

---

## Example 2 :

Input:
s = "Mr Ding"

Output:
"rM gniD"

---

## Algorithm :

1. Define a helper function reverse to reverse a segment of the string s between indices start and end.
2. Initialize two pointers, start and end, to track the beginning and end of each word.
3. Traverse the string:
    - When encountering a space (' ') or the end of the string ('\0'), reverse the word between start and end-1.
    - Update start to the position after the current space and continue traversing.
4. Return the modified string s.

---

## Code :

```
#include <string.h>

void reverse(char* s, int start, int end) {
    char temp;
    while (start <= end) {
        temp = s[start];
        s[start] = s[end];
        s[end] = temp;
        start++;
        end--;
    }
}

char* reverseWords(char* s) {
    int start = 0, end = 0;

    if (strlen(s) < 2) return s;

    int len = strlen(s);

    while (end <= len) {
        if (s[end] == ' ' || s[end] == '\0') {
```

```
            reverse(s, start, end - 1);
            start = end + 1;
        }
        end++;
    }
    return s;
}
```

---

## Time Complexity :

- O(n): The string is traversed once to locate words, and each word is reversed in O(m), where mmm is the word length. The overall complexity is linear with respect to the string length n.

## Space Complexity :

- O(1): The reversal is done in place without requiring additional memory.

---

## Edge Cases :

1. Single word input:
    - Example: s = "word" → Output: "drow".
2. Multiple spaces between words: (not applicable here, as per constraints).
3. String with only one character:
    - Example: s = "a" → Output: "a".
4. Longest input: Handle up to $5 \times 10^4$ characters efficiently.