<https://leetcode.ca/2016-06-03-186-Reverse-Words-in-a-String-II/>

**Description**

Given a character array s, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in s will be separated by a single space.

Your code must solve the problem **in-place,** i.e. without allocating extra space.

**Example 1:**

**Input:** s = ["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"]

**Example 2:**

**Input:** s = ["a"]

**Output:** ["a"]

**Constraints:**

1 <= s.length <= 105

s[i] is an English letter (uppercase or lowercase), digit, or space ' '.

There is **at least one** word in s.

s does not contain leading or trailing spaces.

All the words in s are guaranteed to be separated by a single space.

**Attempt 1: 2025-07-05**

**Solution 1: Two Pointers (10 min)**

class Solution {

public void reverseWords(char[] s) {

int n = s.length;

for (int i = 0, j = 0; j < n; ++j) {

if (s[j] == ' ') {

reverse(s, i, j - 1);

i = j + 1;

} else if (j == n - 1) {

reverse(s, i, j);

}

}

reverse(s, 0, n - 1);

}

private void reverse(char[] s, int i, int j) {

for (; i < j; ++i, --j) {

char t = s[i];

s[i] = s[j];

s[j] = t;

}

}

}

Time Complexity: O(n)

Space Complexity: O(1)

**Refer to Deepseek**

**LeetCode 186. Reverse Words in a String II - Java Solution**

**Problem Statement**

Given a character array s, reverse the order of the words in-place (without allocating extra space for another array). A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space.

**Approach**

**Reverse the entire string**: This puts the words in the correct order but each word is backwards

**Reverse each word individually**: This corrects the word order within each word

**Handle edge cases**: Leading/trailing spaces, multiple spaces between words

**Solution Code**

class Solution {

public void reverseWords(char[] s) {

// Step 1: Reverse the entire array

reverse(s, 0, s.length - 1);

// Step 2: Reverse each word individually

int start = 0;

for (int end = 0; end <= s.length; end++) {

if (end == s.length || s[end] == ' ') {

reverse(s, start, end - 1);

start = end + 1;

}

}

}

private void reverse(char[] s, int left, int right) {

while (left < right) {

char temp = s[left];

s[left] = s[right];

s[right] = temp;

left++;

right--;

}

}

}

**Explanation**

**First Reverse (Whole String)**:

Turns "the sky is blue" into "eulb si yks eht"

Words are now in reverse order but each word is backwards

**Second Reverse (Individual Words)**:

For each word (between spaces or start/end):

Reverse the characters in the word

"eulb" → "blue", "si" → "is", etc.

Final result: "blue is sky the"

**Edge Cases Handled**:

Multiple spaces between words

Leading/trailing spaces

Single word input

**Complexity Analysis**

**Time Complexity**: O(n)

Each character is visited twice (once during full reverse, once during word reverses)

**Space Complexity**: O(1)

All operations are done in-place with constant extra space

This solution efficiently reverses the word order while maintaining the in-place requirement, making it optimal for the problem constraints.

**Refer to**

[L344.Reverse String (Ref.L345,L541)](note://WEBfd25055adcca24c731abdeb888e6e120)