

## Solution: Reverse Words in a String

Let's solve the Reverse Words in a String problem using the Two Pointers pattern.

### We'll cover the following

- Statement
- Solution
  - Time complexity
  - Space complexity

## Statement

Given a sentence, reverse the order of its words without affecting the order of letters within a given word.

### Constraints:

- Sentence contains English uppercase and lowercase letters, digits, and spaces.
- $1 \leq \text{sentence.length} \leq 10^4$
- The order of the letters within a word is not to be reversed.

**Note:** The input string may contain leading or trailing spaces or multiple spaces between words. The returned string, however, should only have a single space separating each word. Do not include any extra spaces.

## Solution

In this problem, we first reverse the complete string. Now take two pointers, `start` and `end`, initialized with the start of the list, which is index `0`.

Now, iterate a loop until `start` is less than the length of the list, and in each iteration, move the `end` pointer forward until it hits a space. At this point, we have a complete word starting from the `start` index to the `end-1` index, but with the characters in reverse order.

To change the order of characters, we call the `strRev` function with the starting and ending positions of the word. This will reverse the characters in the word.

Now, we update the `start` and `end` pointers to the next of `end` pointer, which is basically the first character of the next word. Now, repeat this process for the next word. At the end of all iterations, we get the reversed words in the string.

The following illustration shows these steps in detail:

Here's the sentence that we want to reverse.

0	1	2	3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	---	---	---	----	----

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		F	r	i	e	n	d

1 of 20



We can see the code of this solution below.

Java

```

1  import java.util.*;
2
3  class ReverseWords {
4
5      public static String reverseWords(String s) {
6          // trim spaces and convert string to string builder
7          char[] s1 = s.toCharArray();
8          String s2 = cleanSpaces(s1, s1.length);
9          StringBuilder builder = new StringBuilder(s2);
10
11         // reverse the whole string using the strRev() function
12         strRev(builder, 0, builder.length() - 1);
13
14         // reverse every word
15         int n = builder.length();
16         int start = 0, end = 0;
17         // Find the start index of each word by detecting spaces.
18         while (start < n) {
19             // Find the end index of the word.

```

```

22         // Let's call our helper function to reverse the word in-place.
23         strRev(builder, start, end - 1);
24         // moving to the next word
25         start = end + 1;
26         ++end;
27     }
28

```

Reverse Words in a String

## Time complexity

Because the array is traversed twice, the time complexity of this solution is  $O(n + n) = O(n)$ , where  $n$  is the length of the string.

## Space complexity

The space complexity of this solution is  $O(n)$  as, at the start of the algorithm, to overcome the issue of strings being immutable in Java, we copy it into a list of characters.

← Back

Reverse Words in a St...

Next →

Valid Palindrome II

✓ Completed



