Daily Coding Problem #113

Problem

This problem was asked by Google.

Given a string of words delimited by spaces, reverse the words in string. For example, given "hello world here", return "here world hello"

Follow-up: given a mutable string representation, can you perform this operation in-place?

Solution

One way we can solve this problem is by splitting the input string into a list of words, and then reversing the order.

```
def reverse_words(string):
```

```
    words = string.split(' ')
    words = reversed(words)
    return ' '.join(words)
```

This solution works in `O(N)` time, and uses `O(N)` space since we create a new list to hold the words, and a new string to represent the output.

If we were given a mutable string representation instead of an immutable string, we can perform the reverse operation in-place. Let's assume we are given a list of characters, which we will modify in-place. We can easily reverse a substring within a string in `O(N)` time and `O(1)` space. However, simply reversing the entire string or reversing each word won't get return the correct solution -- we must do both operations. First, we reverse the entire string to get the string "ereh dlrow olleh". Then, we reverse each word within the string to obtain the original words: "here world hello".

```
def reverse_words(string_list):
    # Helper function to reverse string in place
    def reverse(l, start, end):
        # Reverses characters from index start to end (inclusive)
        while start < end:
            l[start], l[end] = l[end], l[start]
            start += 1
            end -=1

    # Reverse the entire string
    reverse(string_list, 0, len(string_list) - 1)

    # Reverse each word in the string
    start = 0
    for end in range(len(string_list)):
        if string_list[end] == ' ':
            print(start, end)
            reverse(string_list, start, end - 1)
```

```
            start = end + 1
    # Reverse the last word
    reverse(string_list, start, len(string_list) - 1)

    return string_list
```

This solution has a worst-case time complexity of `O(N)`, since each character is swapped a maximum of two times. This solution also gives us a worst-case space complexity of `O(1)`.