Problem 186: Reverse Words in a String II

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem 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 <= 10

5

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.

Code Snippets

C++:

```
class Solution {
public:
void reverseWords(vector<char>& s) {

}
};
```

Java:

```
class Solution {
public void reverseWords(char[] s) {

}
}
```

Python3:

```
class Solution:
def reverseWords(self, s: List[str]) -> None:
"""
Do not return anything, modify s in-place instead.
"""
```

Python:

```
class Solution(object):
def reverseWords(self, s):
"""
:type s: List[str]
:rtype: None Do not return anything, modify s in-place instead.
"""
```

JavaScript:

```
/**
* @param {character[]} s
* @return {void} Do not return anything, modify s in-place instead.
*/
var reverseWords = function(s) {

};
```

TypeScript:

```
/**
Do not return anything, modify s in-place instead.
*/
function reverseWords(s: string[]): void {

};
```

C#:

```
public class Solution {
public void ReverseWords(char[] s) {

}
}
```

C:

```
void reverseWords(char* s, int sSize) {

}
```

Go:

```
func reverseWords(s []byte) {

}
```

Kotlin:

```
class Solution {
fun reverseWords(s: CharArray): Unit {

}
}
```

Swift:

```
class Solution {
func reverseWords(_ s: inout [Character]) {

}
}
```

Rust:

```
impl Solution {
pub fn reverse_words(s: &mut Vec<char>) {

}
}
```

Ruby:

```
# @param {Character[]} s
# @return {Void} Do not return anything, modify s in-place instead.
def reverse_words(s)

end
```

PHP:

```
class Solution {

/**
* @param String[] $s
* @return NULL
*/
function reverseWords(&$s) {

}
}
```

Dart:

```
class Solution {
void reverseWords(List<String> s) {

}
}
```

Scala:

```
object Solution {
def reverseWords(s: Array[Char]): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
void reverseWords(vector<char>& s) {

}
};
```

Java Solution:

```
/**
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public void reverseWords(char[] s) {

}
}
```

Python3 Solution:

```
"""
Problem: Reverse Words in a String II
Difficulty: Medium
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def reverseWords(self, s: List[str]) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reverseWords(self, s):
"""
:type s: List[str]
:rtype: None Do not return anything, modify s in-place instead.
"""
```

JavaScript Solution:

```
/**
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {character[]} s
* @return {void} Do not return anything, modify s in-place instead.
*/
var reverseWords = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
Do not return anything, modify s in-place instead.
*/
function reverseWords(s: string[]): void {

};
```

C# Solution:

```
/*
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public void ReverseWords(char[] s) {

}
}
```

C Solution:

```
/*
* Problem: Reverse Words in a String II
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

void reverseWords(char* s, int sSize) {

}
```

Go Solution:

```
// Problem: Reverse Words in a String II
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func reverseWords(s []byte) {

}
```

Kotlin Solution:

```
class Solution {
fun reverseWords(s: CharArray): Unit {

}
}
```

Swift Solution:

```
class Solution {
func reverseWords(_ s: inout [Character]) {

}
}
```

Rust Solution:

```
// Problem: Reverse Words in a String II
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn reverse_words(s: &mut Vec<char>) {

}
```

```
}
```

Ruby Solution:

```
# @param {Character[]} s
# @return {Void} Do not return anything, modify s in-place instead.
def reverse_words(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $s
* @return NULL
*/
function reverseWords(&$s) {

}
}
```

Dart Solution:

```
class Solution {
void reverseWords(List<String> s) {

}
}
```

Scala Solution:

```
object Solution {
def reverseWords(s: Array[Char]): Unit = {

}
}
```