Reverse Words in a String

Given an input string, reverse the string word by word.

For example,
Given s = "`the sky is blue`",
return "`blue is sky the`".

Update (2015-02-12):
For C programmers: Try to solve it *in-place* in *O*(1) space.

click to show clarification.

Clarification:

- What constitutes a word?
  A sequence of non-space characters constitutes a word.
- Could the input string contain leading or trailing spaces?
  Yes. However, your reversed string should not contain leading or trailing spaces.
- How about multiple spaces between two words?
  Reduce them to a single space in the reversed string.

Solution 1

```
class Solution {
public:
    void reverseWords(string &s) {
        string result;
        int pos = 0;
        for (int i = 0; i < s.size(); i ++){
            if (s[i] == ' '){
                if (i > pos )
                    result = s.substr(pos,i-pos)+ " " + result ;
                pos = i + 1;
            }
            else if (i == s.size()-1)
                result = s.substr(pos,s.size()-pos)+" "+result;
        }
        s = result.substr(0,result.size()-1) ;
    }
};
```

written by exodia original link here

Solution 2

First, reverse the whole string, then reverse each word.

```
void reverseWords(string &s) {
    reverse(s.begin(), s.end());
    int storeIndex = 0;
    for (int i = 0; i < s.size(); i++) {
        if (s[i] != ' ') {
            if (storeIndex != 0) s[storeIndex++] = ' ';
            int j = i;
            while (j < s.size() && s[j] != ' ') { s[storeIndex++] = s[j++]; }
            reverse(s.begin() + storeIndex - (j - i), s.begin() + storeIndex);
            i = j;
        }
    }
    s.erase(s.begin() + storeIndex, s.end());
}
```

written by yuruofeifei original link here

Solution 3

```
String[] parts = s.trim().split("\\s+");
String out = "";
if (parts.length > 0) {
    for (int i = parts.length - 1; i > 0; i--) {
        out += parts[i] + " ";
    }
    out += parts[0];
}
return out;
```

I'm splitting on the regex for one-or-more whitespace, this takes care of multiple spaces/tabs/newlines/etc in the input. Since the input could have leading/trailing whitespace, which would result in empty matches, I first trim the input string.

Now there could be three possibilities: 1. The input is empty, return an empty string directly. 2. The input contains only one part, return that part. 3. The input contains multiple parts, reverse the order, making sure we don't end with a trailing space.

Obviously this is not the fastest or most memory efficient way to solve the problem, but optimizations should *only* be done when they are needed. Readable code is usually more important than efficient code.

How to make it efficient?

1. Use a StringBuilder to concatenate the string parts, instead of concatenating strings directly. This will (I assume) build something like a linked-list of string parts, and only allocate the new string when you need it, instead of on each concatenation.
2. Iterate over the string, instead of using trim/split. Store the index of the last character in the word, when you find the first character, copy the substring to the output string.
3. Instead of using substring, insert the word-characters directly in the StringBuilder. Assuming they're using a linked-list or tree, this could be a whole last faster.

written by daniel12 original link here