Reverse Words in a String

Problem Statement: Given a string s, reverse the words of the string.

Examples:

Example 1:

Input: s="this is an amazing program"

Output: "program amazing an is this"

Example 2:

Input: s="This is decent"

Output: "decent is This"

Solution:

Disclaimer: *Don't jump directly to the solution, try it out yourself first.*

Solution 1(Brute Force)

Intuition: We just need to print the words in reverse order. Can we somehow store them in reverse order of the occurrence and then simply add it to our answer?

Approach

- Use a stack to push all the words in a stack
- Now, all the words of the string are present in the stack, but in reverse order
- Pop elements of the stack one by one and add them to our answer variable. Remember to add a space between the words as well.
- Here's a quick demonstration of the same

Code:

- C++ Code
- Java Code

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string s="TUF is great for interview preparation";
    cout<<"Before reversing words: "<<endl;
    cout<<s<<endl;
    s+=" ";
    stack<string> st;
    int i;
    string str="";
    for(i=0;i<s.length();i++)
    {
        if(s[i]==' ')
        {
```

```
            st.push(str);
            str="";
        }
        else str+=s[i];
    }
    string ans="";
    while(st.size()!=1)
    {
        ans+=st.top()+" ";
        st.pop();
    }
    ans+=st.top();// The last word should'nt have a space after it
    cout<<"After reversing words: "<<endl;
    cout<<ans;
    return 0;
}
```

Output:

Before reversing words:
TUF is great for interview preparation
After reversing words:
preparation interview for great is TUF

Time Complexity: O(N), Traversing the entire string

Space Complexity: O(N), Stack and ans variable

Solution 2(Optimized Solution)

Intuition: Notice, that we are using a stack in order to perform our task. Can we somehow not use it and reverse the words as we move through the string? Could we store a word in reverse order when we are adding it to our answer variable?

Approach:

- We start traversing the string from the end until we hit a space. It indicates that we have gone past a word and now we need to store it.

- We check if our answer variable is empty or not
- If it's empty, it indicates that this is the last word we need to print, and hence, there shouldn't be any space after this word.
- If it's empty we add it to our result with a space after it. Here's a quick demonstration of the same

Code:

- C++ Code
- Java Code

```cpp
#include<bits/stdc++.h>
using namespace std;
string result(string s)
{
    int left = 0;
    int right = s.length()-1;

    string temp="";
    string ans="";
```

```
    //Iterate the string and keep on adding to form a word
    //If empty space is encountered then add the current word to the result
    while (left <= right) {
        char ch= s[left];
        if (ch != ' ') {
            temp += ch;
        } else if (ch == ' ') {
            if (ans!="") ans = temp + " " + ans;
            else ans = temp;
            temp = "";
        }
        left++;
    }

    //If not empty string then add to the result(Last word is added)
    if (temp!="") {
        if (ans!="") ans = temp + " " + ans;
        else ans = temp;
    }

    return ans;
}
int main()
{
    string st="TUF is great for interview preparation";
    cout<<"Before reversing words: "<<endl;
    cout<<st<<endl;
    cout<<"After reversing words: "<<endl;
    cout<<result(st);
    return 0;
}
```

Output:

Before reversing words:
TUF is great for interview preparation
After reversing words:
preparation interview for great is TUF

Time Complexity: O(N), N~length of string

Space Complexity: O(1), Constant Space