Brute Force Approach:

Approach:

- Split the string s into individual words.
- Reverse the order of the words.
- Join the words back into a single string with a single space separating them.

```cpp
string reverseWords(string s) {
    vector<string> words;
    string word = "";

    for (char c : s) {
        if (c == ' ') {
            if (!word.empty()) {
                words.push_back(word);
                word = "";
            }
        } else {
            word += c;
        }
    }

    if (!word.empty()) words.push_back(word);

    reverse(words.begin(), words.end());

    string result = "";
    for (int i = 0; i < words.size(); i++) {
        result += words[i];
        if (i < words.size() - 1) result += " ";
    }

    return result;
}
```

Complexity:

- **Time Complexity:** O(n), where n is the length of the string. The string is traversed multiple times, but the operations are linear.
- **Space Complexity:** O(n), to store the words and the final result.

Better Approach:

Approach:

- Trim leading and trailing spaces.
- Split the string s into words and reverse the list of words.
- Join the words back into a single string with a single space.

```cpp
string reverseWords(string s) {
    // Remove leading and trailing spaces
    int left = 0, right = s.size() - 1;
    while (left <= right && s[left] == ' ') left++;
    while (right >= left && s[right] == ' ') right--;

    deque<string> words;
    string word = "";

    while (left <= right) {
        char c = s[left];

        if (c == ' ' && !word.empty()) {
            words.push_front(word);
            word = "";
        } else if (c != ' ') {
            word += c;
        }

        left++;
    }

    words.push_front(word);

    string result = "";
    while (!words.empty()) {
        result += words.front();
        words.pop_front();
        if (!words.empty()) result += " ";
    }

    return result;
}
```

Complexity:

- **Time Complexity:** O(n), where n is the length of the string. The string is processed in linear time.

- **Space Complexity:** O(n), due to the use of the deque and the final result.

Best Approach:

Approach:

- First, remove any extra spaces by iterating through the string once.
- Reverse the entire string.
- Reverse each word individually.

```cpp
string reverseWords(string s) {
    int n = s.size();

    // Remove leading, trailing, and extra spaces
    int i = 0, j = 0;
    while (i < n) {
        while (i < n && s[i] == ' ') i++;
        if (i < n && j > 0) s[j++] = ' ';
        int start = j;
        while (i < n && s[i] != ' ') s[j++] = s[i++];
        reverse(s.begin() + start, s.begin() + j);
    }
    s.resize(j);

    // Reverse the entire string
    reverse(s.begin(), s.end());

    return s;
}
```

Complexity:

- **Time Complexity:** O(n), since each character in the string is visited at most twice.
- **Space Complexity:** O(1), since the reversal is done in-place.