First Unique Character Index In String LeetCode

This C++ program finds the index of the first unique character in a given string

Approach 1: Function to find the index of the first unique character in a string.

1. It initializes an array **count** to count the frequency of each character in the input string (assuming lowercase English letters).
2. It iterates through the input string **s** and increments the count for each corresponding character in the **count** array.
3. Then, it iterates through the input string again and checks if the count of the character at the current position is equal to 1. If so, it returns the index of that character as the index of the first unique character.
4. If no unique character is found, it returns -1.
    - **Time Complexity: O(n), where 'n' is the length of the input string. The code iterates through the string twice, but each iteration is linear in the length of the string.**
    - **Space Complexity: O(1) as the count array has a constant size of 26 (assuming lowercase English letters).**

Approach 2: Function to find the index of the first unique character in a string using a queue.

1. It initializes a queue **q** to keep track of the indices of characters as they are processed.
2. It initializes an array **count** to count the frequency of each character in the input string.
3. It iterates through the input string **s**. For each character, it increments the count of that character in the **count** array and pushes its index onto the queue.
4. After pushing the index, it checks if the count of the character at the front of the queue (the oldest character in the queue) is greater than 1. If yes, it means that this character is no longer unique, so it removes indices of characters with frequency > 1 from the front of the queue.
5. The loop continues until all characters are processed or until the queue is empty.
6. After the loop, if the queue is empty, it means no unique character was found, so it returns -1. Otherwise, it returns the index of the first character in the queue, which is the index of the first unique character.

- **Time Complexity: O(n), where 'n' is the length of the input string. The code iterates through the string once, and each operation on the queue is constant time.**
- **Space Complexity: O(k), where 'k' is the number of unique characters in the input string. In the worst case, all unique characters are pushed onto the queue.**