Find If a String is Valid Palindrome LeetCode

The code aims to determine whether a given string is a palindrome. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward, disregarding spaces, punctuation, and capitalization.

Example: Input: "A man, a plan, a canal: Panama" Output: The String "A man, a plan, a canal: Panama" is a palindrome

Approach 1: Function to check if a string is a palindrome

- The function takes a string as input and creates a reversed version of the string using the **reverseString** function.
- It then iterates through the original string and the reversed string, comparing corresponding characters.
- Non-alphanumeric characters are skipped using the **isalnum** function.
- If any pair of corresponding characters does not match, the function returns **false**.
- If all characters match, the function returns **true**.
- **Time Complexity: checkPalindrome has a time complexity of O(n), where n is the length of the input string. This is because it requires iterating through the string twice: once to create the reversed string and once to compare characters.**
- **Space Complexity: checkPalindrome has a space complexity of O(n), where n is the length of the input string. This is because it creates an additional string of the same length as the input string to store the reversed version.**

Approach 2: Optimized function to check if a string is a palindrome

- The function takes a string as input and uses two pointers, **start** and **end**, initialized at the beginning and end of the string, respectively.
- It iterates through the string, comparing corresponding characters.
- Non-alphanumeric characters are skipped using the **isalnum** function.
- If any pair of corresponding characters does not match, the function returns **false**.
- If all characters match, the function returns **true**.
- **Time Complexity: checkPalindromeOptimized has a time complexity of O(n), where n is the length of the input string. It only requires a single iteration through the string.**

- **Space Complexity: checkPalindromeOptimized has a space complexity of O(1) since it does not require any additional data structures.**

Which approach to use:

- Approach 2 (checkPalindromeOptimized) is recommended as it provides the same functionality with better space efficiency. It avoids the need for creating a separate reversed string, resulting in improved space complexity.