Problem Link:

https://leetcode.com/problems/valid-palindrome/

Problem Description:

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a* ***palindrome****, or* `false` *otherwise*.

Problem Approach:

Two Pointers

Sample Test Case:

Input: s = "A man, a plan, a canal: Panama"

Output: true

Solution:

We start by initializing two pointers, left and right. Then we start a loop that continues as long as left is less than right. It will effectively check characters from both ends of the string towards the center. Then, we have two inner loops which basically skip all the non-alphanumeric characters, from left and right pointer. Then, we check whether the lowercase version of the characters match each other or not. If they don't, no need to check any further, we return False from there only. And if they do, then we move to the next iteration. If the loop continues till the end, then it means that the given String is a palindrome. And that's our answer!

The time complexity of this solution is O(n) and the space complexity is O(1).

Code (Python):

```
def isPalindrome(self, s: str) -> bool:
        left, right = 0, len(s) - 1
        while left < right:
            while left < right and not s[left].isalnum():
                left += 1
            while left < right and not s[right].isalnum():
                right -= 1
            if s[left].lower() != s[right].lower():
                return False
            left += 1
            right -= 1
        return True
```