[Click on the logo to find **Problem Statement**]

Intuition

The `isPalindrome` function employs a two-pointer approach to check whether a given string `s` is a palindrome, considering only alphanumeric characters and ignoring case. The two pointers, `left` and `right`, start at the beginning and end of the string, respectively, and move towards the center. The function skips non-alphanumeric characters and compares the alphanumeric characters at the pointers, disregarding case. If at any point the characters don't match, the string is not a palindrome, and the function returns False. This process continues until the pointers meet or cross each other, and if no mismatches are found, the function returns True, indicating that the string is a palindrome. The use of a helper function, `alphanum`, determines whether a character is alphanumeric.

Approach

1. **Initialize Pointers:**
    - Initialize two pointers, `left` and `right`, at the beginning and end of the input string `s` respectively.
2. **Two-Pointer Traverse:**
    - Use a while loop to traverse the string from both ends towards the center (`left < right`).
    - Skip non-alphanumeric characters by incrementing `left` and decrementing `right` until valid characters are found.
3. **Comparison:**
    - Compare the alphanumeric characters at positions `s[left]` and `s[right]`, ignoring case.
    - If they do not match, return False, indicating that the string is not a palindrome.
4. **Move Pointers:**
    - Increment `left` and decrement `right` to move the pointers closer to the center.
5. **Palindrome Check:**
    - Repeat steps 2-4 until the pointers meet or cross each other.

6. **Return Result:**
    - If no mismatches are found during traversal, return True, indicating that the string is a palindrome.
7. **Alphanumeric Check:**
    - The `alphanum` helper function checks whether a character is alphanumeric.

Complexity

- Time complexity: O(n)
- Space complexity: O(1)

Code

```
class Solution:
    def isPalindrome(self, s: str) -> bool:
        left, right = 0, len(s) - 1

        while left < right:
            while left < right and not self.alphanum(s[left]):
                left += 1
            while right > left and not self.alphanum(s[right]):
                right -= 1
            if s[left].lower() != s[right].lower():
                return False
            left,right = left + 1, right - 1
        return True

    def alphanum(self, c):
        return(ord('A') <= ord(c) <= ord('Z') or
               ord('a') <= ord(c) <= ord('z') or
               ord('0') <= ord(c) <= ord('9'))
```