125. Valid Palindrome

Solved

Easy | Topics | Companies

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.*

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

Example 3:

> **Input:** s = " "
> **Output:** true
> **Explanation:** s is an empty string "" after removing non-alphanumeric characters.
> Since an empty string reads the same forward and backward, it is a palindrome.

Constraints:

- `1 <= s.length <= 2 * 10^5`
- `s` consists only of printable ASCII characters.

Python:

```
class Solution:
    def isPalindrome(self, s: str) -> bool:
        # Step 1: Filter out non-alphanumeric characters and convert to lowercase
        filtered = [ch.lower() for ch in s if ch.isalnum()]

        # Step 2: Check if filtered string equals its reverse
        return filtered == filtered[::-1]
```

JavaScript:

```
/**
 * @param {string} s
 * @return {boolean}
 */
var isPalindrome = function(s) {
    // Step 1: convert to lowercase and remove non-alphanumeric chars
    s = s.toLowerCase().replace(/[^a-z0-9]/g, "");

    // Step 2: two-pointer check
    let left = 0;
    let right = s.length - 1;

    while (left < right) {
```

```
        if (s[left] !== s[right]) {
            return false;
        }
        left++;
        right--;
    }

    return true;
};
```

Java:

```
class Solution {
    public boolean isPalindrome(String s) {
        if (s == null || s.length() == 0) {
            return true;
        }

        int left = 0, right = s.length() - 1;

        while (left < right) {
            // Move left pointer until alphanumeric
            while (left < right && !Character.isLetterOrDigit(s.charAt(left))) {
                left++;
            }

            // Move right pointer until alphanumeric
            while (left < right && !Character.isLetterOrDigit(s.charAt(right))) {
                right--;
            }

            // Compare after converting both to lowercase
            if (Character.toLowerCase(s.charAt(left)) != Character.toLowerCase(s.charAt(right))) {
                return false;
            }

            left++;
            right--;
        }

        return true;
    }
}
```