

Valid Palindrome

A string that reads the same forwards and backwards

● String

■ EASY

What You Will Learn

- Two-pointer technique: start from both ends, meet in middle
- How to skip non-alphanumeric characters
- Case-insensitive comparison with `toLowerCase()`
- Why this is a gateway problem for two-pointer mastery

Intuition First

'A man, a plan, a canal: Panama' — ignore spaces and punctuation, lowercase everything → 'amanaplanacanalpanama'. Now use two fingers: one at start, one at end. Move them toward each other. If every pair of letters matches, it's a palindrome.

```
Input: 'A man, a plan, a canal: Panama'
```

```
Cleaned: 'amanaplanacanalpanama'
```

```
Two pointers:
```

```
L→ a m a n a p l a n a c a n a l p a n a m a ←R
```

```
a == a ✓ → move both inward
```

```
m == m ✓ → move both inward
```

```
a == a ✓ ...
```

```
... L meets R → PALINDROME ✓
```

```
With punctuation (skip non-alphanumeric):
```

```
'race a car' → 'raceacar' → r≠r... wait
```

```
r==r ✓, a==a ✓, c==a ✗ → NOT palindrome
```

Java Code

```
1 class Solution {
2     public boolean isPalindrome(String s) {
3         int left = 0, right = s.length() - 1;
4
5         while (left < right) {
6             // Skip non-alphanumeric from left
7             while (left < right && !Character.isLetterOrDigit(s.charAt(left)))
8                 left++;
9             // Skip non-alphanumeric from right
10            while (left < right && !Character.isLetterOrDigit(s.charAt(right)))
11                right--;
```

```

12
13     // Compare case-insensitively
14     if (Character.toLowerCase(s.charAt(left)) !=
15         Character.toLowerCase(s.charAt(right)))
16         return false;
17
18     left++; right--;
19 }
20 return true;
21 }
22 }

```

■ Complexity

Type	Value	Meaning
Time	$O(n)$	Each character visited at most once
Space	$O(1)$	Only two pointer variables

■ Common Mistakes

■ Mistake

Using `s.toLowerCase().replaceAll("[^a-z0-9]", "")` first — valid but uses $O(n)$ extra space. The two-pointer approach skips cleaning, so it's $O(1)$ space.

■ Quick Revision

✓	Two pointers: left starts at 0, right at end
✓	Skip non-alphanumeric with <code>isLetterOrDigit()</code>
✓	Compare with <code>toLowerCase()</code> to ignore case
✓	If any mismatch → return false
✓	Time $O(n)$, Space $O(1)$