Palindrome(Sentence)

2nd Review	@October 28, 2025
# Attempts	2
Date Solved	@October 27, 2025
Difficulty	Easy
Status	Solved
Topic/Pattern	Two Pointers

LINK → https://neetcode.io/problems/is-palindrome?list=neetcode150

Problem

- Given a string `s`, determine if it is a **palindrome**, meaning it reads the same forward and backward.
- You should ignore cases, spaces, and punctuation marks (only consider alphanumeric characters).

Example

Input	Output	Reason
`"Was it a car or a cat I saw"`	`True`	After cleaning → `"wasitacaroracatisaw"`, which is same reversed
`"A man, a plan, a canal: Panama"`	`True`	After cleaning → `"amanaplanacanalpanama"`
`"hello"`	`False`	`"hello"` ≠ `"olleh"`

Approaches

1. **Brute Force — Reverse and Compare**

```
def is_palindrome(s: str) → bool:
    s = ''.join(c.lower() for c in s if c.isalnum())  # keep letters/digits only
    return s == s[::-1]  # compare string with its reverse
```

- **Time:** O(n) → cleaning + reversing
- **Space:** O(n) → new cleaned and reversed strings
- **Notes:** Very readable and simple, but creates extra strings in memory.

---

2. **Optimized Two-Pointer Approach**

```
def is_palindrome(s: str) → bool:
    s = ''.join(c.lower() for c in s if c.isalnum())
    i, j = 0, len(s) - 1

    while i <= j:
        if s[i] != s[j]:
            return False
        i += 1
        j -= 1
    return True
```

- **Time:** O(n) → each character checked at most once
- **Space:** O(1) → constant extra space (ignoring cleaned string creation)
- **Notes:** Efficient and avoids creating a reversed copy.
- **Concepts Used:** Two-pointer technique, string normalization, character comparison.

Summary

Approach	Time	Space	Notes
Brute Force	O(n)	O(n)	Simple but creates extra reversed string
Two-Pointer	O(n)	O(1)	Efficient, memory-friendly

Edge Cases

Input	Output	Reason
`""`	`True`	Empty string is symmetric
`"a"`	`True`	Single character is always a palindrome
`" "`	`True`	Only spaces → treated as empty
`"0P"`	`False`	`"0p"` ≠ `"p0"`

Mistakes

- .strip() only removes the spaces from the fronr and the end. It does not remove space from the between the words in a line.

Tip

- Use `.isalnum()` to ignore punctuation and spaces efficiently.
- Always convert to lowercase to make it **case-insensitive**.
- The **two-pointer approach** is preferred for interviews because it’s clear and optimal.
- A good follow-up variation: check for **almost palindromes** (allow at most one mismatch)