four LeetCode problems (125, 344, 680, 977) with:

- **Problem link**
- **Intuition / Approach**
- **Why using this approach**
- **Time & Space Complexity**
- **Step-by-step explanation**
- **Mind map / key points**

? LeetCode Problem Solutions – Two Pointer Series

This repository contains professional explanations and solutions for the following **classic two-pointer problems**:

Problem	Link
125 – Valid Palindrome	[LeetCode 125]https://leetcode.com/problems/valid-palindrome/description/
344 – Reverse String	LeetCode 344
680 – Valid Palindrome II	LeetCode 680
977 – Squares of a Sorted Array	LeetCode 977

1? LeetCode 125 – Valid Palindrome

Problem: Check if a string is a palindrome **ignoring case and non-alphanumeric characters.**

Intuition:

- Palindrome → string reads same forward and backward
- Use **two pointers** (start, end)
- Skip non-alphanumeric characters
- Convert to lowercase to handle case-insensitive comparison
- Compare characters at both pointers → move inward

Why Two Pointers:

- Efficient → only one pass
- No extra memory needed

Approach / Steps:

1. Initialize `start=0`, `end=n-1`
2. Skip characters if not alphanumeric

3. Compare lowercase characters
4. Mismatch → return false
5. Loop ends → return true

Code:

```
string s = "A man, a plan, a canal: Panama";
int start = 0, end = s.size() - 1;

while (start < end) {
    if (!isalnum(s[start])) { start++; continue; }
    if (!isalnum(s[end])) { end--; continue; }
    if (tolower(s[start]) != tolower(s[end])) {
        cout << "false";
        return 0;
    }
    start++;
    end--;
}
cout << "true";
```

Time Complexity: O(n) → scan each character once **Space Complexity:** O(1) → in-place comparison

---

2️⃣ LeetCode 344 – Reverse String

Problem: Reverse a character array **in-place** with O(1) extra memory

Intuition:

- Swap first and last element, move pointers inward until they meet
- Use **two pointers** (`start` and `end`)

Why Two Pointers:

- In-place swap → no extra array required
- Single pass → O(n)

Approach / Steps:

1. Initialize `start=0`, `end=n-1`
2. Swap `s[start]` and `s[end]`
3. Move `start++`, `end--`
4. Repeat until start >= end

Code:

```
vector<char> s = {'h','e','l','l','o'};
int start = 0, end = s.size() - 1;
```

```
while (start < end) {
    swap(s[start], s[end]);
    start++;
    end--;
}
```

Time Complexity: O(n) **Space Complexity:** O(1)

3⃣ LeetCode 680 – Valid Palindrome II

Problem: Determine if a string can become a palindrome **by deleting at most one character**

Intuition:

- Two pointers check for mismatch
- If mismatch → two options: remove left or remove right
- If either option results in palindrome → true

Why This Approach:

- Efficient → O(n)
- Only one pass
- Handles “at most one deletion” constraint

Steps:

1. Initialize `start=0`, `end=n-1`
2. While `start < end`
   - If match → move inward
   - If mismatch → check two options: `start+1,end` OR `start,end-1`
3. If any option returns true → palindrome possible

Code:

```
bool check(vector<char>& s, int start, int end) {
    while (start < end) {
        if (s[start] != s[end]) return false;
        start++; end--;
    }
    return true;
}

int main() {
    vector<char> s = {'a','b','c','a'};
    int start=0,end=s.size()-1;
    while (start<end) {
        if (s[start]==s[end]) { start++; end--; }
        else {
            cout << (check(s,start+1,end) || check(s,start,end-1));
            return 0;
        }
    }
    cout << "true";
}
```

Time Complexity: O(n) → single pass + one extra check **Space Complexity:** O(1)

4️⃣ LeetCode 977 – Squares of a Sorted Array

Problem: Given a **sorted array**, return **squares of each number** in **sorted order**

Intuition:

- Negative numbers when squared may be largest
- Use **two pointers** at start and end
- Compare squares → fill result array from end to start

Why This Approach:

- Single pass → O(n)
- Efficient for sorted array

Steps:

1. Initialize `start=0`, `end=n-1`, `k=n-1`
2. Compare `nums[start]^2` vs `nums[end]^2`
3. Place larger at `ans[k]`, k–
4. Move pointers inward
5. Repeat until start > end

Code:

```
class Solution {
public:
    vector<int> sortedSquares(vector<int>& nums) {
        int n = nums.size();
        int start=0, end=n-1, k=n-1;
        vector<int> ans(n);

        while (start <= end) {
            int leftSq = nums[start]*nums[start];
            int rightSq = nums[end]*nums[end];
            if (leftSq > rightSq) {
                ans[k] = leftSq;
                start++;
            } else {
                ans[k] = rightSq;
                end--;
            }
            k--;
        }
        return ans;
    }
};
```

Time Complexity: O(n) **Space Complexity:** O(n)

▯ Mind Map / Quick Revision (All Problems)

```
125 – Valid Palindrome
  Two pointer → skip non-alpha → lowercase → compare

344 – Reverse String
  Two pointer → swap start/end → in-place

680 – Valid Palindrome II
  Two pointer → mismatch → delete left or right → check palindrome

977 – Squares of Sorted Array
  Two pointer → compare squares → fill from end
```

✅Why Two Pointer Approach Works for All:

- Efficient → O(n) scan
- Simple logic for start/end comparison
- Handles in-place operations
- Minimal extra memory

–