

Day 2 — **Two-Pointer Patterns **



All Problems Solved Using Two-Pointer Approach — Colorful VSCode-Styled C++ Code

Table of Contents

1. Two-Pointer Pattern — Quick Recipe
2. Problems (All Two-Pointer Based)
 - 2.1 Valid Palindrome
 - 2.2 Reverse String
 - 2.3 Squares of a Sorted Array
 - 2.4 Valid Palindrome II
 - 2.5 Valid Word Abbreviation
3. Tips & Pitfalls
4. Quick Summary Sheet

1 Two-Pointer Pattern — Quick Recipe

Where to Use: Arrays, Strings, Linked Lists (Linear Data)

How it Works:

- Start two pointers: `i = 0` (left), `j = n - 1` (right)
- Move both based on logic or condition
- Continue until `i >= j`

General Template:

```
int i = 0, j = n - 1;
while (i < j) {
    // check condition
    // move pointers
}
 Time: O(n) | Space: O(1)
```

2□ Problems – All Using Two-Pointer

2.1 Valid Palindrome

Why Two-Pointer? Need to compare both ends `for` equality.

Logic:

`i → start, j → end`

Ignore non-alphanumeric characters

Compare lowercase values

Mismatch → **false**, else continue

C++ Code:

```
cpp
Copy code
#include <bits/stdc++.h>
using namespace std;

bool isPalindrome(string s) {
    int i = 0, j = s.size() - 1;
    while (i < j) {
        if (!isalnum(s[i])) { i++; continue; }
        if (!isalnum(s[j])) { j--; continue; }
        if (tolower(s[i]) != tolower(s[j])) return false;
        i++; j--;
    }
    return true;
}
```

✓ Example: "A man, a plan, a canal: Panama" → **true**

2.2 Reverse String (In-Place)

Why Two-Pointer? Swap both ends till middle.

Logic:

i = 0, j = n-1

Swap characters

Move both pointers toward center

C++ Code:

```
cpp
Copy code
#include <bits/stdc++.h>
using namespace std;

void reverseString(vector<char>& s) {
    int i = 0, j = s.size() - 1;
    while (i < j) {
        swap(s[i], s[j]);
        i++; j--;
    }
}
```

✓ Example: [h, e, l, l, o] → [o, l, l, e, h]

2.3 Squares of a Sorted Array

Why Two-Pointer? Largest squares appear at both ends.

Logic:

```
i = 0, j = n-1, k = n-1

Compare abs(nums[i]) & abs(nums[j])
```

Bigger square → store at res[k--]

C++ Code:

```
cpp
Copy code
#include <bits/stdc++.h>
using namespace std;

vector<int> sortedSquares(vector<int>& nums) {
    int n = nums.size();
    vector<int> res(n);
    int i = 0, j = n - 1, k = n - 1;
    while (i <= j) {
        if (abs(nums[i]) > abs(nums[j])) {
            res[k--] = nums[i] * nums[i];
            i++;
        } else {
            res[k--] = nums[j] * nums[j];
            j--;
        }
    }
    return res;
}
```

Example: [-4, -1, 0, 3, 10] → [0, 1, 9, 16, 100]

2.4 ✎ Valid Palindrome II

Why Two-Pointer? Compare both ends, allow one deletion.

Logic:

Normal palindrome check

On mismatch → try skipping left once OR right once

C++ Code:

```
cpp
Copy code
#include <bits/stdc++.h>
using namespace std;

bool isPal(string &s, int i, int j) {
    while (i < j) {
        if (s[i] != s[j]) return false;
        i++; j--;
    }
    return true;
}
```

```

bool validPalindrome(string s) {
    int i = 0, j = s.size() - 1;
    while (i < j) {
        if (s[i] != s[j]) {
            return isPal(s, i + 1, j) || isPal(s, i, j - 1);
        }
        i++; j--;
    }
    return true;
}

```

✓ Example: abca → remove 'b' → true

2.5 Valid Word Abbreviation

Why Two-Pointer? Move through both strings together.

Logic:

i on word, j on abbr

If abbr[j] is letter → match

If digit → read full number, skip in word

Both must end together

C++ Code:

cpp

Copy code

```
#include <bits/stdc++.h>
using namespace std;
```

```

bool validWordAbbreviation(string word, string abbr) {
    int i = 0, j = 0;
    while (i < word.size() && j < abbr.size()) {
        if (isdigit(abbr[j])) {
            if (abbr[j] == '0') return false;
            int num = 0;
            while (j < abbr.size() && isdigit(abbr[j])) {
                num = num * 10 + (abbr[j] - '0');
                j++;
            }
            i += num;
        } else {
            if (word[i] != abbr[j]) return false;
            i++; j++;
        }
    }
    return i == word.size() && j == abbr.size();
}
```

✓ Example: word = "international", abbr = "i121" → true

3 □ Tips & Pitfalls

✓ Always explain why two-pointer suits the problem.

⚠ Use `static_cast<unsigned char>` with `isalnum` safely.

✓ Dry run examples (interviewer loves clarity).

✓ Mention time & space complexity.

✓ Don't forget bounds check when skipping indices.

4 Quick Summary Sheet ⏳

Problem Approach Time Space

Valid Palindrome Compare ends skipping non-alnum $O(n)$ $O(1)$

Reverse String Swap till middle $O(n)$ $O(1)$

Squares Array Compare `abs` ends, fill from back $O(n)$ $O(n)$

Valid Palindrome II Skip one `char` on mismatch $O(n)$ $O(1)$

Word Abbreviation Move both pointers $O(n + m)$ $O(1)$

🌟 Final Tip: Har problem me two-pointer ka reason clearly likho aur example ke sath explain karo – interviewer ko lagega tum pattern-based soch rakhte ho ✎