# Day 2 — Two-Pointer Patterns 🧠✨

> All problems solved using Two-Pointer Approach — Simple Logic + Easy Code 💡

---

## 🧭 Table of Contents

---

## 1️⃣ Two-pointer pattern — quick recipe 🍳

**Where use:**

- Array, String, Linked list (linear data)

**How works:**

- Start two pointers: `i = 0` (left), `j = n-1` (right)
- Move both based on 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)

---

## 2️⃣ Problems — all using TWO-POINTER 🔥

### 2.1 ✅ Valid Palindrome

**Why two-pointer?** Need to check both ends for equality.

**Logic:**

1. `i` → start, `j` → end
2. Ignore non-alphanumeric
3. Compare lowercase characters
4. If mismatch → false, else continue

**C++ 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:**

1. `i=0` , `j=n-1`
2. Swap characters
3. Move both pointers toward center

**C++ 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 are at both ends.

**Logic:**

1. `i=0` , `j=n-1` , `k=n-1`
2. Compare abs(nums[i]) & abs(nums[j])
3. Bigger square → put in res[k--]

**C++ 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 ends, allow one deletion on mismatch.

**Logic:**

1. Normal palindrome check
2. On mismatch → skip left once OR right once and recheck

**C++ 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 along word & abbreviation together.

**Logic:**

1. `i` on word, `j` on abbr
2. If abbr[j] is letter → match
3. If digit → read number → skip those many chars in word
4. At end both pointers must finish together

**C++ 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 = “i12l” → true

## 3️⃣ Tips & Pitfalls 📌

- ✅ Always state why two-pointer fits.
- ⚠️ Use `static_cast<unsigned char>` with `isalnum`, `isdigit` safely.
- ✅ Dry run examples — interviewer loves visual clarity.
- ✅ Complexity mention karo har code ke end me.
- ✅ `i += num` ke baad bounds check karna na bhoolo.

## 4️⃣ 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 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 logic clearly mention karo — kyun use kar rahe ho aur kaise move karte ho. Isse interviewer ko lagega ki tum pattern-based thinker ho. 🚀