Two-Pointer & Sliding Window Cheat Sheet

1️⃣ Array Two-Pointer Problems

Problem	Key Idea / Pattern	LeetCode
Container With Most Water	Move smaller pointer inward to maximize area	11
3Sum	Sort + fix one element + two pointers for remaining sum	15
3Sum Closest	Similar to 3Sum, track closest sum	16
4Sum	Sort + nested loops + two pointers	18
Two Sum II (sorted array)	Two pointers moving inward to find sum	167
Valid Triangle Number	Sort + largest side fixed + two pointers for pairs	611
Squares of a Sorted Array	Two pointers from both ends to fill new array	977
Remove Duplicates from Sorted Array	Slow & fast pointers	26
Move Zeroes	Two pointers to swap zero/non-zero	283
Partition Array by Parity	Separate even/odd numbers	905

2️⃣ String Two-Pointer Problems

Problem	Key Idea / Pattern	LeetCode
Valid Palindrome	Left & right pointers moving inward	125
Reverse String / Words	Swap characters using two pointers	344 / 151
Minimum Window Substring	Sliding window + two pointers	76
Permutation in String	Sliding window + character count	567
Longest Substring Without Repeating Characters	Sliding window	3
Backspace String Compare	Two pointers from end	844

3️⃣ Sliding Window / Subarray Problems (Two-Pointer Variant)

Problem	Key Idea	LeetCode
Maximum Sum Subarray of Size K	Window sum	209 (small variant)
Subarray Product Less Than K	Right pointer expands, left contracts	713
Longest Substring with At Most K Distinct Characters	Sliding window	340
Count Subarrays with K Different Integers	Sliding + hash map	992
Longest Repeating Character Replacement	Sliding window	424

4️⃣ Patterns to Remember

1. **Opposite ends (start/end)** → max/min problems like Container With Most Water.
2. **Sliding window** → substring/subarray sums, counts, or conditions.
3. **Fixed + two-pointer** → 3Sum, 4Sum, valid triangle.
4. **Slow & fast / write pointer** → in-place array modifications.
5. **Matching / comparing two sequences** → isSubsequence, backspace string compare.

5️⃣ Key Code Snippets

Two-pointer on array:

```
int l = 0, r = arr.size() - 1;
while (l < r) {
    // process arr[l] and arr[r]
    if (arr[l] < arr[r]) l++;
    else r--;
}
```

Sliding window on string/array:

```
int left = 0;
for (int right = 0; right < s.size(); right++) {
    // include s[right] in window
    while (window invalid) left++;
```

```
    // process valid window
}
```

Slow & fast / write pointer for in-place array:

```
int slow = 0;
for (int fast = 0; fast < arr.size(); fast++) {
    if (arr[fast] != 0) arr[slow++] = arr[fast];
}
// fill remaining positions if needed
```

Tip: Most interviewers test **sorting + two pointers** or **sliding window**. Practice these patterns with multiple variations for maximum prep.