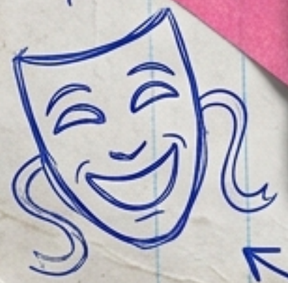
My DSA Study Notes!

Topic: The Two Pointer Technique.



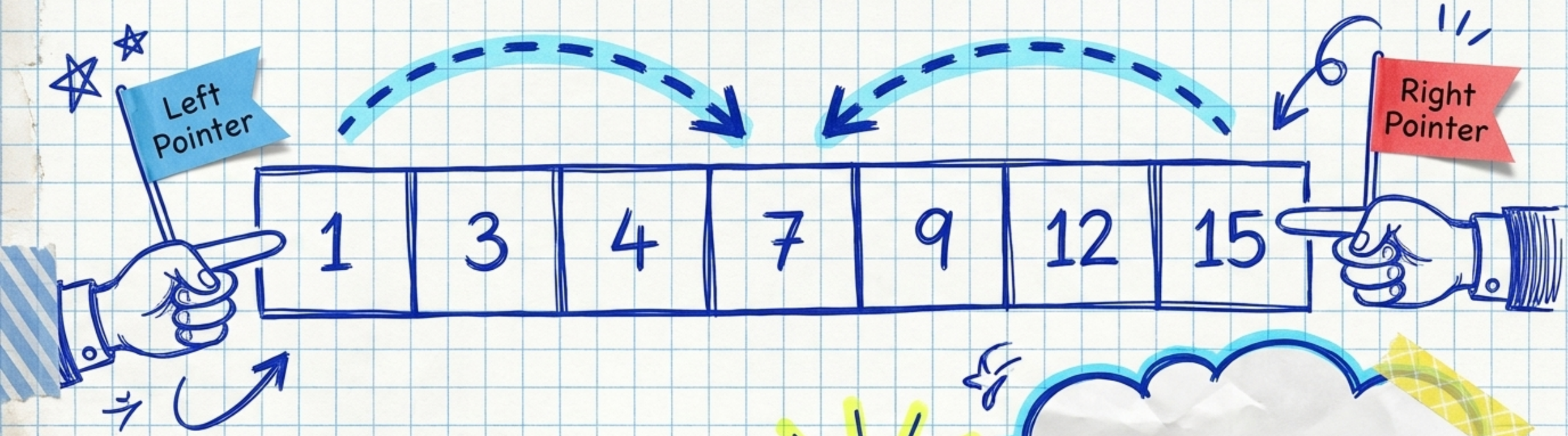
Goal: Stop using Brute Force!
Learn to spot the pattern instead of just memorizing code.

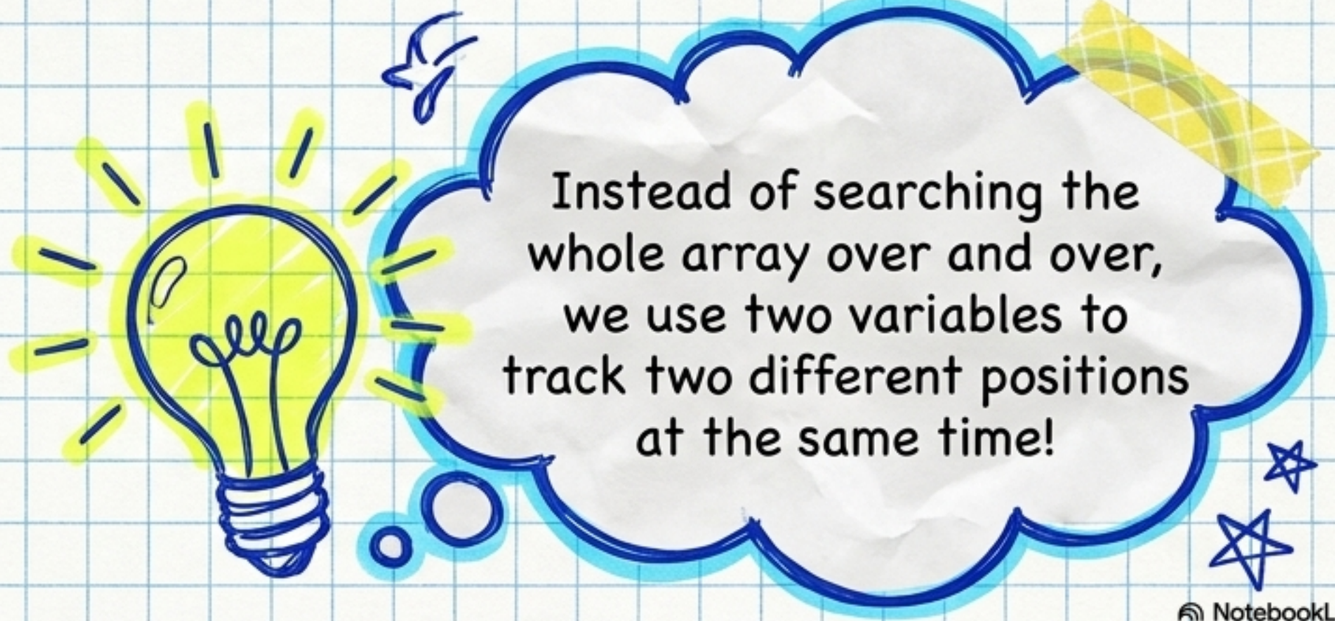
The Syllabus: Roadmap to 2026

- [x] Lecture 1: Complete Syllabus & Roadmap
- [x] Lecture 2: Two Pointers Introduction
- [ ] Lectures 3-5: Problem 1 - Pair with Target Sum
- [ ] Lectures 6-7: Problem 2 - Remove All Occurrences

Next!

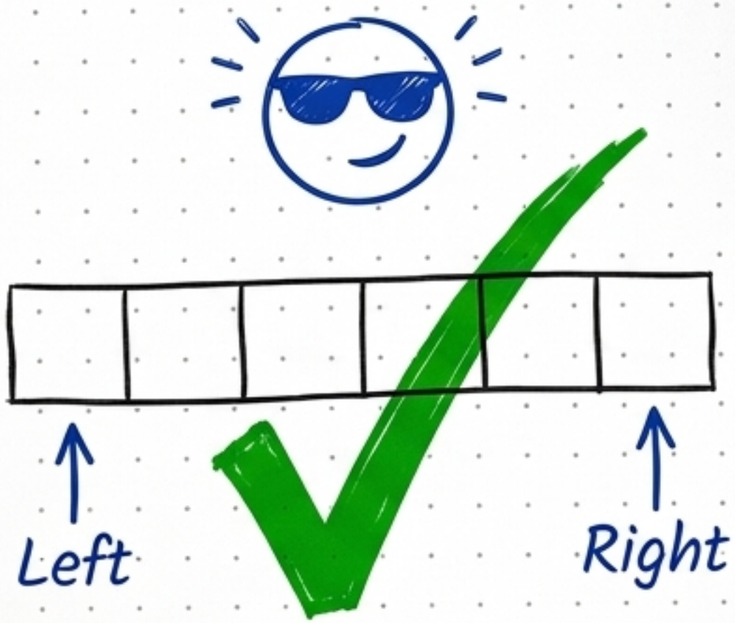
What actually is the Two Pointer Technique?





Why are we learning this? (Brute Force vs. Two Pointers)

Nested Loops. $O(N^2)$ Time.
Very slow.
Fails the interview.



Two independent variables.
O(N) Time. Fast!
Passes the interview.

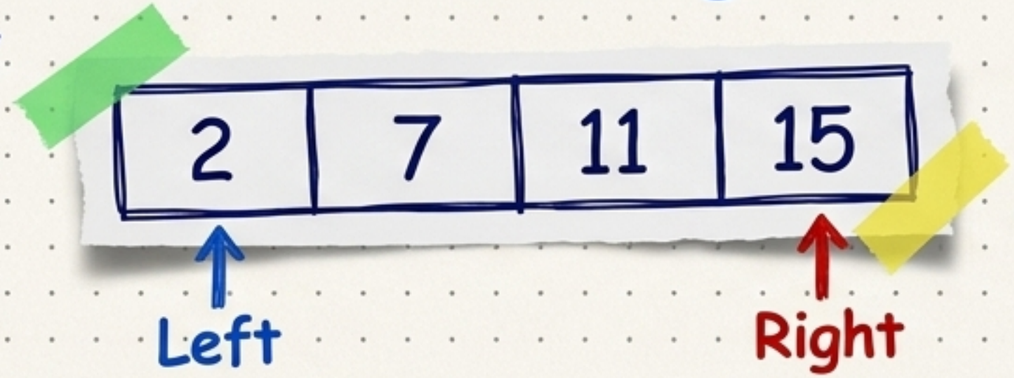
Problem 1: Pair with Target Sum

Given a sorted array of integers, find two numbers that add up to a specific target sum. Return their indices.

Hint: The fact that it's SORTED is the dead giveaway to use Two Pointers!

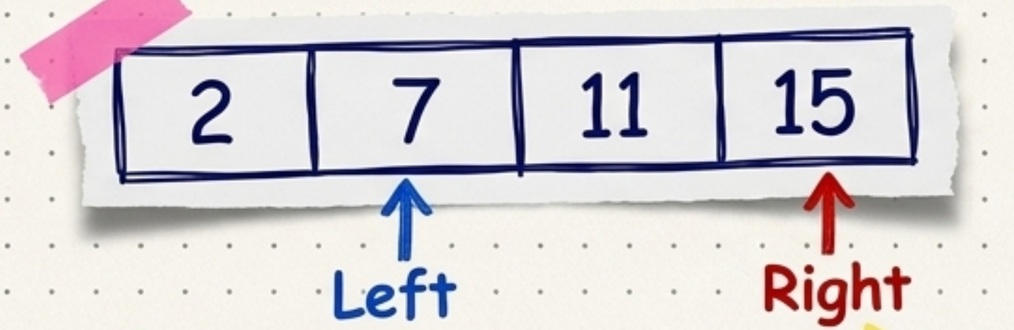
How to solve: Pointers moving towards each other

Target = 18



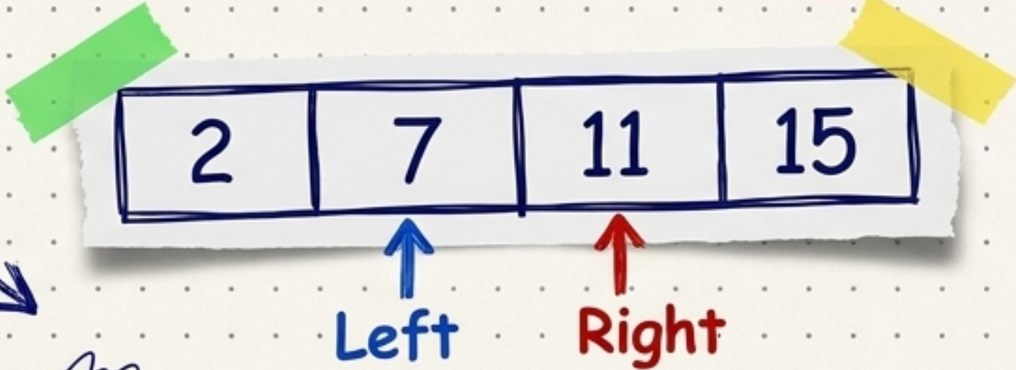
2 + 15 = 17. Too small!
Move Left pointer up to get a bigger number.

bigger number



7 + 15 = 22. Too big!
Move Right pointer down to get a smaller number.

smaller number



7 + 11 = 18.
Perfect Match!
We found the pair!

The Code: Pair with Target Sum

```
int left = 0;
int right = nums.length - 1;
while (left < right) {
    int sum = nums[left] + nums[right];
    if (sum == target) return new int[]{left, right};
    if (sum < target) left++;
    else right--;
}
```

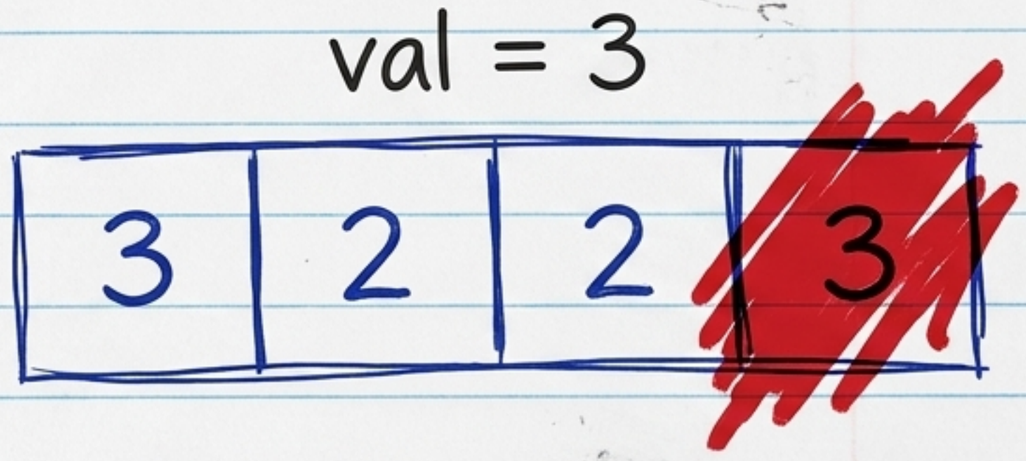
Keep going until they crash into each other!

Too small? Increment left.

Too big? Decrement right.

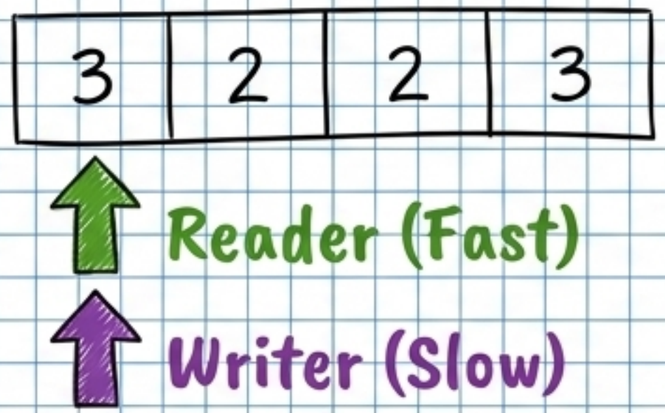
Problem 2: Remove All Occurrences of an Element

Given an array and a value, remove all instances of that value in-place and return the new length of the array.



Wait... the array isn't sorted this time. And we are deleting things? How do two pointers. How do two pointers help here?

How to solve: Pointers moving in the SAME direction



Step 1:
Reader sees 3
(the target).

---->

Trash!
Skip it.

(Writer stays put)

---->

Step 2:
Reader sees 2.

--->

Keep this!

(Writer writes 2 into its current spot, then Writer moves forward)

Rule: One pointer acts as a scout looking for good numbers. The other pointer acts as the builder, placing the good numbers at the front!

The Code: Remove Occurrences

```
int slow = 0;
for (int fast = 0; fast < nums.length; fast++) {
    if (nums[fast] != val) {
        nums[slow] = nums[fast];
        slow++;
    }
}
return slow;
```

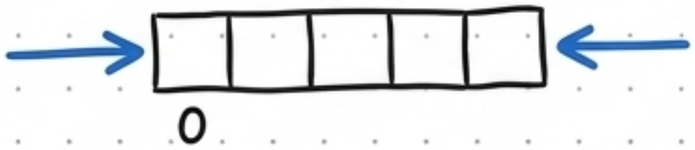
Our Writer pointer starts at index 0.

Did the Reader find a safe number?

If yes, copy it over and move the Writer forward!

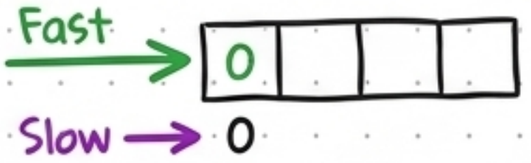
Study Cheat Sheet: The Two Pointer Rulebook

Opposite Ends



- Best for: Sorted arrays, finding pairs, target sums.
- Example: Pair with Target Sum.

Same Direction (Fast/Slow)



- Best for: In-place modifications, removing duplicates, skipping elements.
- Example: Remove Occurrences.

Master these two patterns, and you've mastered 80% of Two Pointer interview questions! Ready for 2026!