1. #1 Two Sum
   Pattern: Hash map / complement lookup
   Recognition: “Find two numbers that add to target” with indices preserved → brute force screams TLE bait.
   Core idea: As you scan, ask “what number would complete this pair?” and remember what you’ve already seen.
   Trick: Store value → index, not index → value.
   Perk: Works even with negatives; order doesn’t matter; single pass possible.
2. #26 Remove Duplicates from Sorted Array
   Pattern: Two pointers (slow–fast overwrite)
   Recognition: Sorted array + “in-place” + “unique count”.
   Core idea: Slow pointer marks last unique; fast scans. When different, advance slow and overwrite.
   Trick: You don’t delete—just overwrite and return length.
   Perk: Final garbage values don’t matter at all.
3. #27 Remove Element
   Pattern: Two pointers / overwrite filter
   Recognition: “Remove all occurrences of x in-place”.
   Core idea: Keep a write index; copy only valid elements.
   Trick: Order doesn’t matter → you can also swap with last for O(1) extra.
   Perk: Same mental model as #26, but condition-based instead of uniqueness-based.
4. #66 Plus One
   Pattern: Carry propagation
   Recognition: Array represents number; add 1; watch for 9s.
   Core idea: Walk from end, manage carry, stop early if possible.
   Trick: If all digits are 9, prepend 1.
   Perk: Teaches “don’t convert to int” instinct.
5. #88 Merge Sorted Array
   Pattern: Reverse two pointers
   Recognition: Two sorted arrays, first has extra space.
   Core idea: Fill from the back to avoid overwriting.
   Trick: Always compare from ends, write at k--.
   Perk: Reverse thinking saves extra memory.
6. #121 Best Time to Buy and Sell Stock
   Pattern: Running minimum / greedy
   Recognition: Max difference with buy before sell.

Core idea: Track minimum so far; update profit greedily.

Trick: Never explicitly choose buy/sell days—profit emerges naturally.

Perk: Prototype for many “best window so far” problems.

7. #169 Majority Element

   Pattern: Boyer–Moore Voting

   Recognition: “Element appears more than n/2 times”.

   Core idea: Cancel out different elements; survivor is majority.

   Trick: Count up/down instead of frequency map.

   Perk: O(1) space, feels magical but mathematically solid.

8. #238 Product of Array Except Self

   Pattern: Prefix × Suffix

   Recognition: “Except self” + division forbidden.

   Core idea: Left product pass + right product pass.

   Trick: Reuse output array for prefix, carry suffix in variable.

   Perk: Classic demonstration of space optimization.

9. #53 Maximum Subarray

   Pattern: Kadane’s Algorithm

   Recognition: Max sum of contiguous subarray.

   Core idea: Either extend current subarray or restart at current element.

   Trick: Local best vs global best mindset.

   Perk: Appears everywhere in disguised forms.

10. #334 Increasing Triplet Subsequence

    Pattern: Greedy with two sentinels

    Recognition: “Exists i < j < k such that nums[i] < nums[j] < nums[k]”.

    Core idea: Track smallest and second smallest so far.

    Trick: You don’t need indices, just values.

    Perk: Teaches “existence proof” thinking instead of construction.

11. #189 Rotate Array

    Pattern: Reverse trick

    Recognition: Rotate right by k, in-place.

    Core idea: Reverse whole array, then reverse first k and rest.

    Trick: k %= n always.

    Perk: Reveals how reversing is a structural weapon.

12. #42 Trapping Rain Water

    Pattern: Prefix max + Suffix max (or two pointers)

    Recognition: Histogram bars + water trapped between heights.

Core idea: Water at i = min(max left, max right) − height[i].

Trick: Precompute left_max and right_max or compress into two pointers.

Perk: Once internalized, this pattern unlocks skyline, container, and elevation problems.

Meta-memory hook:
If the problem smells like “ignore order → overwrite”, think two pointers.
If it says “except self”, think prefix/suffix.
If it says “best so far”, think greedy.
If it says “exists”, not “find”, think minimal state tracking.
If it says “water / skyline / histogram”, think boundaries before values.

These problems aren’t isolated puzzles—they’re a compressed vocabulary. Once the pattern clicks, the code is just handwriting practice.