🧠 The 5-Step Mantra to Approach ANY Coding Problem

Memorize this. Use it **every single time**.

🪜 STEP 1: Classify the Problem (NOT solve it)

> *"What family does this belong to?"*

Before touching examples or dry runs, ask:

- Array? → Two pointers / Sliding window / Prefix
- Subarray? → Equality vs inequality
- Stack behavior? → Monotonic?
- Matrix? → Simulation / index-sum?
- Optimization? → Binary search on answer?
- Counting pairs? → Merge sort?

🧠 **Your brain blanks because it tries to solve instead of classify.**

🪜 STEP 2: Ask ONE Killer Question

> *"What property stays consistent? What relationship or rule solves this?"*

Examples:

- Subarray sum → prefix difference
- Majority → cancellation

- Inversion → sorted halves
- NGE → monotonic order
- Diagonal → index sum (i+j)
- Sliding window → monotonic expand/shrink

Every problem has **one invariant or the core logic/rule**.

🪜 STEP 3: Decide BRUTE FORCE FIRST

> *“If I had infinite time, what would I do?”*

This removes fear.

Examples:

- Check all subarrays
- Check all pairs
- Simulate literally

Then ask:

> *“Which part is repeated?”*

That repeated work becomes:

- Hashing
- Two pointers
- Stack
- Merge sort

🪜 STEP 4: Choose ONE TEMPLATE/ DATA STRUCTURE

> *“Which template reduces repeated work? What do I need to remember from the past?”*

Never invent logic.

Pick from:

- Two pointers
- Sliding window
- Prefix sum
- Monotonic stack
- Merge sort
- BFS/DFS
- Greedy

If none fit → the problem is probably **simulation**.

Write:

DS = ____, **stores** = ____

🪜 STEP 4: Define LOOP/ TRAVERSAL

Ask:

> *“Am I traversing once or nested?”*

- ☐ Single loop
- ☐ Two pointers
- ☐ DFS recursion
- ☐ BFS level-wise

Write the loop idea in **one line**.

🪜 STEP 5: Dry Run ONLY AFTER Template

Dry run is for **validation**, not discovery.

If you dry run without a template:
❌ You memorize
❌ You panic
❌ You blank next time

If you dry run after the template:
✅ You confirm
✅ You adapt
✅ You retain

🔥 THE ONE-LINE MANTRA (Write this!)

> **Classify Pattern → Find invariant rule → Think brute → Choose template/data structure → Define loop/ traversal → Dry run**

Say it **out loud** in practice.