

Playbook

Interview Coding Playbook: Solve 2 Hard Problems in 45 Minutes

THE 7-STEP FRAMEWORK

STEP 1: Clarify & Own the Problem (1-2 min)

Don't touch your keyboard yet.

Ask about:

- Input limits and ranges?
- Empty inputs possible?
- Duplicates allowed?
- Hidden constraints?

Confirm:

- Does order matter?
- Is input sorted?
- Always valid answer, or return special value?

Goal: Spot hidden traps before coding.

STEP 2: Manual Walkthrough - Find the Pattern (2-3 min)

Do this:

- Work through **2+ edge cases by hand**
- Write it out physically: mark indices, highlight subarrays, underline repeats
- Explain out loud like teaching a friend

Why: Most bugs come from skipping this. Interviewers watch HOW you approach cases.

STEP 3: Brute Force First (2-3 min)

State the naive solution:

- Substring search? Try every start/end
- Grid traversal? Check all placements
- **Write out time/space complexity and say it aloud**

Why:

- Proves you can reason with structure
 - Acts as correctness check for optimized solution
 - Don't apologize for $O(N^2)$
-

STEP 4: Find the Bottleneck (1-2 min)

Look for repeated work:

- Rescanning same substring?
- Recalculating same map/set?
- Revisiting same cells?

Say out loud: "Here's where time blows up."

Ask: Can I carry over work from previous step/window/row?

This reveals: Sliding window, two pointers, DP, BFS/DFS, prefix sums

STEP 5: Choose & Justify Your Pattern (1 min)

State the technique name:

- "Using sliding window because I need continuous segment with X property"
- "This is BFS because I need shortest distance"
- "Hash map gives $O(1)$ lookups for counts"

Defend your choice:

- "Can't sort because order matters"

- "No backtracking needed—unique answer exists"

Interviewers notice: Can you defend WHY, not just WHAT?

STEP 6: Dry Run Your Logic (3-4 min)

Take one non-trivial test case:

- Track EVERY pointer, map entry, queue operation
- Step through: window expansion/contraction, count updates, match detection

Look for:

- Missing updates
- Off-by-one errors
- Empty inputs
- Duplicate keys

If you find bugs: Fix on paper, THEN code.

STEP 7: Write Clean Code (Rest of time)

Before main loop:

- Declare all variables
- Write small, readable functions if possible

Handle carefully:

- HashMap missing keys
- Counter increments
- Edge cases (empty input, not found)

Debug smartly:

- Use print statements for dry runs
- Don't "wing it" on Hard problems

Time yourself in practice: Track what slows you down—logic, coding, or debugging?

MINDSET & ROUTINE

- ✓ Interviews reward calm, repeatable problem-solving—not genius.
 - ✓ Never skip Steps 1-2. Always do final dry run.
 - ✓ Your structure pulls you through even on bad days.
 - ✓ Speed comes from process, not rushing.
-

QUICK REFERENCE CHECKLIST

- ☐ Clarified inputs, constraints, edge cases
 - ☐ Manual walkthrough of 2+ examples
 - ☐ Stated brute force with complexity
 - ☐ Identified bottleneck
 - ☐ Named and justified pattern
 - ☐ Dry run with tracking
 - ☐ Clean code with edge case handling
 - ☐ Final verification run
-

Remember: The "magic" is in never skipping steps. Even if time runs short, partial completion with clear reasoning beats rushed, buggy code.