Algorithm Templates Study Guide

Companion to Python Algorithm Templates Cheatsheet

Overview

This study guide accompanies the *Python Algorithm Templates* cheatsheet (25 patterns). Use this document to:

- Diagnose your weak spots and prioritize study time
- Track progress with spaced repetition
- Map patterns to practice problems
- Prepare systematically for Staff-level interviews at Google/Meta/FAANG

1 Study Strategy: The Weakness-First Approach

1.1 Why Not Sequential?

Don't go through patterns 1-25 in order. Here's why:

- **Pareto Principle violation**: You waste time on patterns you already know (Two Pointers, Binary Search) and rush through hard ones (Union-Find, Advanced DP)
- **Fatigue effect**: By pattern #18, you're tired and the hardest patterns get the least energy
- **False confidence**: You breeze through easy patterns, then panic on rusty ones in the interview

1.2 The Optimal Strategy: Attack Weaknesses First

Phase 1: Diagnostic (Day 1 - 20 minutes)
Quickly scan all 25 patterns and categorize:

- **Priority 1 (Red)** : Rusty/Unknown - haven't used recently or never learned well
- **Priority 2 (Yellow)** : Shaky - recognize it, but would struggle with details
- **Priority 3 (Green)** : Confident - could write from memory in an interview right now

Phase 2: Intensive Practice (Days 2-7)
Focus 80% of time on Priority 1 & 2 patterns. For each rusty pattern:

1. **Study the template** (5 min) - understand the pattern deeply
2. **Write from memory** (10 min) - close cheatsheet, try to recreate
3. **Solve 2-3 LeetCode problems** (45 min) - apply the pattern
4. **Review mistakes** (10 min) - what did you forget?
5. **Repeat next day** - spaced repetition cements learning

Phase 3: Sequential Review (Days 8-10)
Now go through patterns 1-25 sequentially:

- 2-3 minutes per pattern
- Verify you can recall the key template
- Catches any gaps and reinforces connections

Phase 4: Pattern Recognition (Days 11-14)
Practice identifying which pattern to use:

1. Read LeetCode problem description
2. **Identify the pattern** (2-3 min) - use Pattern Recognition Guide
3. Solve using the template
4. This is the actual interview skill

1.3 Two-Week Intensive Schedule

Week 1: Attack Weaknesses

- Day 1: Diagnostic + Start Priority 1 patterns
- Day 2-3: Priority 1 patterns (3-4 patterns/day)
- Day 4-5: Priority 2 patterns (4-5 patterns/day)
- Day 6-7: Mixed practice - random problems from Priority 1&2

Week 2: Polish & Pattern Recognition

- Day 8: Sequential review (patterns 1-12)
- Day 9: Sequential review (patterns 13-25)
- Day 10: Timed practice - 3 LC problems under time pressure
- Day 11-12: Pattern recognition drills
- Day 13-14: Mock interviews or hard LC problems

1.4 Spaced Repetition for Rusty Patterns

For each Priority 1 () pattern, follow this schedule:

- **Day 1**: Study + 3 problems + take detailed notes
- **Day 2**: Review notes + 2 problems
- **Day 4**: Quick review (10 min) + 1 problem
- **Day 7**: Quick review (10 min) + 1 problem
- **Day 14**: Final review (5 min)

This neurologically cements the pattern better than 10 hours on Day 1.

2 Common Weak Spots

Most engineers struggle with these patterns (start here if unsure):

- **Union-Find** - path compression & union by rank optimizations are tricky
- **Topological Sort** - must know both Kahn's (BFS) and DFS versions
- **Trie** - forget the TrieNode structure and how to traverse
- **Advanced DP** - state machines and DP on trees require practice
- **Bellman-Ford** - newer addition, less commonly practiced
- **Monotonic Stack** - the increasing/decreasing logic is subtle
- **LCA** - recursive logic requires clear mental model

3 Progress Tracking Table

Use this table to track your study progress. Fill in dates and check boxes as you complete each phase.

Priority Legend:

- **Priority 1 (P1)** : Rusty/Unknown - Master these first with spaced repetition
- **Priority 2 (P2)** : Shaky - Solidify with practice
- **Priority 3 (P3)** : Confident - Quick review sufficient

Note: On B&W printers, priorities show as: P1 (darkest gray), P2 (medium gray), P3 (lightest gray)

Pattern	Pri	D1	D2	D4	D7	D14	Notes / Problems
1. Two Pointers	P3	☐	☐	☐	☐	☐	
2. Sliding Window	P3	☐	☐	☐	☐	☐	
3. Binary Search	P3	☐	☐	☐	☐	☐	
4. Fast/Slow Pointers	P3	☐	☐	☐	☐	☐	
5. Linked List Reversal	P2	☐	☐	☐	☐	☐	
6. Binary Tree Traversal	P3	☐	☐	☐	☐	☐	
7. DFS	P3	☐	☐	☐	☐	☐	
8. BFS	P3	☐	☐	☐	☐	☐	
9. Dynamic Programming	P2	☐	☐	☐	☐	☐	
10. Backtracking	P2	☐	☐	☐	☐	☐	
11. Bit Manipulation	P2	☐	☐	☐	☐	☐	
12. Prefix Sum	P2	☐	☐	☐	☐	☐	
13. Heaps	P2	☐	☐	☐	☐	☐	
14. Monotonic Stack	P1	☐	☐	☐	☐	☐	
15. Overlapping Intervals	P2	☐	☐	☐	☐	☐	
16. Trie	P1	☐	☐	☐	☐	☐	
17. Union-Find	P1	☐	☐	☐	☐	☐	
18. Greedy Algorithms	P2	☐	☐	☐	☐	☐	
19. Advanced DP	P1	☐	☐	☐	☐	☐	
20. Graph Algorithms	P2	☐	☐	☐	☐	☐	
21. Topological Sort	P1	☐	☐	☐	☐	☐	
22. Cyclic Sort	P2	☐	☐	☐	☐	☐	
23. LCA	P1	☐	☐	☐	☐	☐	
24. Matrix Traversal	P2	☐	☐	☐	☐	☐	
25. Bellman-Ford	P1	☐	☐	☐	☐	☐	

Instructions:

1. **Day 1**: Customize priority based on your self-assessment (P1 = rusty, P2 = shaky, P3 = confident)
2. Check boxes as you complete each review session
3. Write problems solved and key insights in the Notes column
4. Focus 80% of time on P1/P2 patterns

4 Pattern-to-LeetCode Problem Mapping

Essential problems for each pattern. Solve these to master the template.

4.1 Priority 1 Patterns (Master These First)

14. Monotonic Stack

- LC 739 - Daily Temperatures (Easy - start here)
- LC 496 - Next Greater Element I (Easy)
- LC 84 - Largest Rectangle in Histogram (Hard - classic)
- LC 42 - Trapping Rain Water (Hard)

16. Trie

- LC 208 - Implement Trie (Medium - must know)
- LC 211 - Design Add and Search Words Data Structure (Medium)
- LC 212 - Word Search II (Hard - combines Trie + DFS)

17. Union-Find

- LC 547 - Number of Provinces (Medium - start here)
- LC 684 - Redundant Connection (Medium)
- LC 323 - Number of Connected Components (Medium)
- LC 1319 - Number of Operations to Make Network Connected (Medium)

19. Advanced DP

- LC 72 - Edit Distance (Medium - 2D DP)
- LC 309 - Best Time to Buy and Sell Stock with Cooldown (Medium - state machine)
- LC 337 - House Robber III (Medium - DP on trees)
- LC 1000 - Minimum Cost to Merge Stones (Hard)

21. Topological Sort

- LC 207 - Course Schedule (Medium - detection)
- LC 210 - Course Schedule II (Medium - ordering)
- LC 269 - Alien Dictionary (Hard)
- LC 310 - Minimum Height Trees (Medium)

23. Lowest Common Ancestor

- LC 236 - Lowest Common Ancestor of a Binary Tree (Medium - must know)
- LC 235 - Lowest Common Ancestor of a BST (Easy - optimization)
- LC 1644 - Lowest Common Ancestor of a Binary Tree II (Medium)

25. Bellman-Ford

- LC 787 - Cheapest Flights Within K Stops (Medium)
- LC 743 - Network Delay Time (Medium - compare with Dijkstra)

4.2 Priority 2 Patterns (Solidify These)

5. Linked List Reversal

- LC 206 - Reverse Linked List (Easy - fundamental)
- LC 92 - Reverse Linked List II (Medium)
- LC 25 - Reverse Nodes in k-Group (Hard)

9. Dynamic Programming

- LC 70 - Climbing Stairs (Easy - start here)
- LC 198 - House Robber (Medium)
- LC 322 - Coin Change (Medium - unbounded knapsack)
- LC 53 - Maximum Subarray (Easy - Kadane's)

10. Backtracking

- LC 46 - Permutations (Medium)
- LC 78 - Subsets (Medium)
- LC 77 - Combinations (Medium)
- LC 79 - Word Search (Medium)

12. Prefix Sum

- LC 560 - Subarray Sum Equals K (Medium - classic)
- LC 974 - Subarray Sums Divisible by K (Medium)
- LC 304 - Range Sum Query 2D (Medium)

13. Heaps / Top K Elements

- LC 215 - Kth Largest Element (Medium)
- LC 347 - Top K Frequent Elements (Medium)
- LC 23 - Merge k Sorted Lists (Hard)
- LC 295 - Find Median from Data Stream (Hard)

15. Overlapping Intervals

- LC 56 - Merge Intervals (Medium - fundamental)
- LC 57 - Insert Interval (Medium)
- LC 435 - Non-overlapping Intervals (Medium)
- LC 252 - Meeting Rooms (Easy)

20. Graph Algorithms

- LC 743 - Network Delay Time (Medium - Dijkstra)
- LC 1334 - Find the City With Smallest Number of Neighbors (Medium - Floyd-Warshall)
- LC 1584 - Min Cost to Connect All Points (Medium - MST)

22. Cyclic Sort

- LC 268 - Missing Number (Easy)

- LC 287 - Find the Duplicate Number (Medium)
- LC 442 - Find All Duplicates in an Array (Medium)

24. Matrix Traversal

- LC 54 - Spiral Matrix (Medium - fundamental)
- LC 48 - Rotate Image (Medium - in-place rotation)
- LC 59 - Spiral Matrix II (Medium)

4.3 Priority 3 Patterns (Quick Review)

1. Two Pointers

- LC 167 - Two Sum II (Easy)
- LC 15 - 3Sum (Medium)

2. Sliding Window

- LC 3 - Longest Substring Without Repeating Characters (Medium)
- LC 424 - Longest Repeating Character Replacement (Medium)

3. Binary Search

- LC 704 - Binary Search (Easy)
- LC 34 - Find First and Last Position (Medium)
- LC 410 - Split Array Largest Sum (Hard - search space)

4. Fast/Slow Pointers

- LC 141 - Linked List Cycle (Easy)
- LC 142 - Linked List Cycle II (Medium)
- LC 876 - Middle of the Linked List (Easy)

7. DFS / 8. BFS

- LC 200 - Number of Islands (Medium)
- LC 994 - Rotting Oranges (Medium - BFS)
- LC 133 - Clone Graph (Medium)

5 Staff-Level Interview Preparation

5.1 What This Cheatsheet Covers

Algorithmic/DSA Portion: 9.5/10 ✓

- All 25 core patterns cover 98%+ of coding interview problems
- Includes advanced patterns (Union-Find, Topological Sort, Advanced DP, Bellman-Ford)
- Optimization techniques (Kadane's, Monotonic Stack, Cyclic Sort)
- Comprehensive graph algorithms

At Staff level, algorithmic complexity doesn't necessarily increase. You'll still see LC Medium/Hard. What changes is the **depth of understanding** and **communication** expected.

5.2 What Staff Interviews Add (Beyond Algorithms)

1. System Design (50% of Staff interviews)

- This cheatsheet: Does not cover
- You need: Scalability, CAP theorem, distributed systems, load balancing, caching
- Resources: *Designing Data-Intensive Applications*, System Design Interview Vol 1 & 2

2. Code Quality & Production Readiness

- Write production-quality code with error handling, edge cases, clean naming
- Discuss time/space tradeoffs explicitly
- Code that "could ship tomorrow"

3. Communication & Problem-Solving Process

- Articulate multiple approaches before coding
- Justify design decisions
- Discuss scalability implications
- Handle ambiguous requirements
- Drive the conversation

4. Optimization & Follow-ups

- "What if we have 1 billion items?"
- "What if this needs to run in real-time?"
- Discuss: Parallelization, memory constraints, hardware limitations

5.3 Pro Tips for Staff-Level Prep

1. **Don't just memorize - understand WHY**
   - Why does Union-Find need path compression?
   - Why does Bellman-Ford relax edges n-1 times?
   - Staff interviewers will ask "why" questions
2. **Practice explaining while coding**
   - Talk through your thought process

- "I'm using a monotonic stack here because..."
- Staff interviews heavily weight communication

3. **Focus on optimization tradeoffs**
   - Know when to use DFS vs BFS
   - Know when Bellman-Ford beats Dijkstra
   - This is where Staff candidates differentiate
4. **The "blank page test"**
   - For your top 5 rusty patterns: can you write them on a blank page from memory?
   - If not, you'll struggle under interview pressure

5.4 Interview Simulation Checklist

Use this for mock interviews or final prep:

☐ Can explain approach before coding (2-3 min)

☐ Can identify pattern within 1-2 minutes

☐ Can write solution from memory without cheatsheet

☐ Can analyze time/space complexity correctly

☐ Can discuss alternative approaches and tradeoffs

☐ Can handle follow-up questions (scalability, edge cases)

☐ Can explain WHY this approach is optimal

☐ Code is clean with good variable names

☐ Communicate clearly throughout the process

☐ Can solve under time pressure (35-40 min for LC Medium)

5.5 Final Week Checklist

☐ Completed all Priority 1 patterns with 3+ problems each

☐ Reviewed all 25 patterns sequentially at least once

☐ Practiced 10+ pattern recognition drills

☐ Completed 3+ full mock interviews

☐ Can write top 10 patterns from memory

☐ Prepared behavioral/leadership STAR stories (5-7 examples)

☐ Reviewed system design fundamentals (if applicable)

☐ Got 8 hours sleep before interview day

Good Luck!

Remember: **This cheatsheet + deliberate practice = Staff-level algorithmic mastery**

The difference between passing and failing isn't knowing more algorithms—it's knowing these 25 patterns **deeply** and recognizing when to use them **instantly**.

Focus on your weak spots. Practice with purpose. Communicate clearly. You've got this!