ChatGPT

LeetCode Pattern-Based DSA Prep (1-Month Plan)

To maximize learning in a short time, focus on *coding patterns* rather than random problems [1] [2]. Each pattern (e.g. two pointers, sliding window, DFS/BFS) recurs across many questions, so mastering it lets you solve dozens of variants efficiently [1] [2]. For example, the *sliding-window* pattern handles contiguous subarray problems [3], while *two-pointer* techniques solve sorted-array pair sums or find midpoints in lists [4]. Likewise, depth-first (DFS) or breadth-first search (BFS) traverse trees/graphs for connectivity and path problems [5]. This targeted approach is far more efficient than solving hundreds of random problems [1] [6].

Key Patterns & Topics

- **Core Data Structures:** Ensure you have basics down – arrays, strings, hash-maps, linked lists, stacks, queues, trees, and graphs [7] [8]. These underlie all patterns.
- **Two Pointers:** Use two indices to process arrays/strings from both ends or at different speeds. Great for problems like sorted-array sum, pair sums, or finding middles/cycles in linked lists [4] [9]. For instance, “Two Sum II (sorted)” or palindrome checks use this pattern.
- **Sliding Window:** Maintain a variable-size window over an array or string to answer range/sum problems [3]. This optimizes contiguous subarray or substring queries (e.g. maximum subarray sum, longest substring without repeats) without re-scanning the entire array each time [3].
- **Fast & Slow Pointers:** A variant of two-pointers where one pointer moves faster. Commonly used to detect cycles in linked lists or to find mid-points [10]. For example, “Linked List Cycle” uses this to see if the fast pointer ever meets the slow one [10].
- **Recursion / Backtracking (DFS):** Explore all possibilities by recursion (depth-first). Key for tree/graph traversals and exhaustive search (e.g. generating permutations, solving N-Queens). Practice basic tree DFS (in-order, pre-order, post-order) and simple backtracking to enumerate combinations.
- **Breadth-First Search (BFS):** Level-by-level traversal using a queue. Useful for shortest-path in unweighted graphs or level-order tree problems [5]. E.g. “Binary Tree Level Order Traversal” and “Minimum Depth of Binary Tree” are BFS staples.
- **Monotonic Stack:** Maintain a stack with elements in sorted order (increasing or decreasing). Solve “next greater/smaller element” problems efficiently [11]. For example, “Next Greater Element” or “Daily Temperatures” use this pattern to quickly find the next larger item.
- **Heaps / Top-K Elements:** Use a priority queue to track extreme values. Practice problems like “Kth Largest Element” or “Merge K Sorted Lists,” where keeping a heap of size K yields the answer [12].
- **Merge Intervals:** For interval or scheduling problems, first sort intervals by start time, then merge overlapping ones. Classic examples include “Merge Intervals” or “Insert Interval” [13].
- **Binary Search:** Apply on sorted data or when solving for a condition threshold. E.g. find an element in a rotated sorted array or search boundaries. The divide-and-conquer search runs in O(log n) on sorted inputs [14].
- **Dynamic Programming (Basics):** Solve optimization/counting problems by breaking into subproblems. Cover simple DP subpatterns like Fibonacci, 0/1 knapsack, subset sum, and longest increasing subsequence [15]. Useful practice problems include “Climbing Stairs,” “House Robber,” and

"Longest Common Subsequence." (Full DP mastery takes longer, so focus on understanding memoization vs tabulation for medium-level DP tasks.)

Each of the above patterns unlocks a variety of LeetCode problems [16] [6]. The *most impactful* 7–10 patterns (two pointers, sliding window, DFS, BFS, binary search, backtracking, basic DP, etc.) cover a huge portion of interview questions [6] [16].

4-Week Study Plan

Divide your 1-month schedule into focused weekly goals. Treat each problem like a mini-interview (write code on paper/editor and create test cases) [17].

- **Week 1 – Arrays & Strings (Warm-up):** Review array and string basics. Practice prefix-sum and hashing techniques (to answer range-sum queries and lookups) and simple sorting or two-pointer cases. Start with easy-medium problems like *Two Sum, Contains Duplicate, Best Time to Buy/Sell Stock, Valid Anagram*, and *Maximum Subarray* [18]. This builds familiarity with array manipulation and simple patterns [18].
- **Week 2 – Linear Data Structures:** Focus on linked lists, sliding-window, and simple matrix problems [19]. Examples: reverse a linked list, detect a cycle in a linked list, and related operations [19]. Solve sliding-window problems on strings/arrays (e.g. *Longest Substring Without Repeating* [20]) and basic matrix/graph problems like *Number of Islands* (which uses DFS/BFS on a grid) [20]. Work on "Container With Most Water" and "Minimum Window Substring" to sharpen two-pointer and sliding techniques (as suggested in week 2 list [21]).
- **Week 3 – Non-Linear Structures (Trees/Graphs):** Practice common tree and graph algorithms [22]. Start with easy-medium tree problems: invert/flip a binary tree, validate BST, maximum depth, and basic traversals (inorder/preorder). Solve *Course Schedule* or *Clone Graph* for directed/undirected graph DFS/BFS, and interval problems like *Merge Intervals* [23]. Also try heap/priority-queue problems like *Top K Frequent Elements* [23]. By week's end you should be comfortable traversing trees/graphs (DFS and BFS) and merging/searching in sorted intervals.
- **Week 4 – Advanced Topics & Review:** Use remaining time to cover anything missed and consolidate. If time permits, tackle tries or hash-based string problems (e.g. *Implement Trie*), and basic DP problems (Climbing Stairs, Coin Change) [15]. Otherwise, review and re-solve questions from previous weeks, and simulate full coding rounds. Revisit the **Blind 75** or similar lists of high-frequency problems [18] to check off gaps. Treat each practice problem as a real interview: write code on paper or a whiteboard, test edge cases, and time yourself [17] [24].

Strategy & Tips

- **Simulate Interview Conditions:** Time yourself and code without reference. Before submitting solutions on LeetCode, run through custom test cases to ensure correctness [17].
- **Learn from Mistakes:** After solving a problem (or struggling through it), review solutions or discussion posts to understand alternate approaches. Leverage resources like *LeetCode Discuss* or blogs to see pattern explanations.
- **Focus on Patterns, Not Memorization:** Always ask "which pattern does this problem fit?" and apply the core idea. As experts note, learning patterns avoids burnout and "teaches you how to recognize the underlying structure" of problems [25] [26].

- **Use Structured Resources (Optional):** If you find it hard to organize, consider pattern-based courses or guides (e.g. *Grokking the Coding Interview: Patterns* on Educative [24] or AlgoMonster by ex-Google engineers) to practice categorized problems. These reinforce the pattern methodology with examples.
- **Balance Breadth and Depth:** Since interviews focus on easy-medium DSA, prioritize mastering patterns over exotic algorithms. According to experienced interviewers, DP is "unavoidable" only for certain companies (e.g. Google) [27]. Cover basic DP only if time permits.

By following this pattern-driven, structured plan, you'll cover the most important DSA topics in one month and build the confidence to handle typical coding interviews. Focus on understanding each pattern conceptually, then practice several LeetCode problems that use that pattern [16] [6]. Good luck!

Sources: Authoritative guides and interview prep blogs discussing LeetCode patterns and study plans [1] [6] [18] [19] [3], among others.

---

[1] [2] [4] [5] [6] [8] [13] [14] [16] [24] [25] [26] 10 Top LeetCode Patterns to Crack FAANG Coding Interviews
https://www.designgurus.io/blog/top-lc-patterns

[3] [10] [11] [12] [15] LeetCode was HARD until I Learned these 15 Patterns
https://blog.algomaster.io/p/15-leetcode-patterns

[7] github.com-seanprashad-leetcode-patterns_-_2021-06-06_16-18-42 : seanprashad : Free Download, Borrow, and Streaming : Internet Archive
https://archive.org/details/github.com-seanprashad-leetcode-patterns_-_2021-06-06_16-18-42

[9] Coding Interviews were HARD, until I learned these Patterns - DEV Community
https://dev.to/somadevtoo/coding-interviews-was-hard-until-i-learned-these-patterns-2ji7

[17] [18] [19] [20] [21] [22] [23] [27] Best practice questions by the author of Blind 75 | Tech Interview Handbook
https://www.techinterviewhandbook.org/best-practice-questions/