



LEARN COMPLETE DSA FOR FREE

WITH MOST ASKED INTERVIEW QUESTIONS

CURATED BY

RAUNAK YADUSH
BHAVESH ARORA



LeetCode



Crack Top Tech Roles with This 70-Day DSA Mastery Roadmap!

Are you preparing for your dream job in tech—whether as a Software Engineer, Data Analyst, or Machine Learning Engineer?

The competition is fierce—but here's the secret weapon that helped me stand out and crack American Express:

A structured 70-Day DSA Roadmap 

Whether you're a beginner or brushing up before interviews, this plan takes you from the fundamentals to advanced problem solving with real LeetCode problems.

What's Inside?

- Topic-wise learning: Arrays, Hashing, Sliding Window, Trees, Graphs, DP & more
- Day-by-day plan with problem links and explanations
- LeetCode questions curated by topic and difficulty
- Ideal for Data & Software roles
- Tracker included for consistency and accountability
- Concepts + Patterns + Practice = Interview-Ready Confidence

Curated by

RAUNAK YADUSH

Software Engineer

2+ Years of Experience |

Linkedin: <https://www.linkedin.com/in/raunakyadush/>

BHAVESH ARORA

Senior Data Analyst at Delight Learning Services

M.Tech - IIT Jodhpur | 3+ Years of Experience

Linkedin: www.linkedin.com/in/bhavesh-arora-11b0a319b

Connect with us: https://topmate.io/bhavesh_arora/

Let's embark on this journey together and make your dreams a reality, starting today.

Week 1: Mastering Arrays (and Time Complexity Basics)

Day 1: Time & Space Complexity

- **Goal:** Learn Big-O, Big-Ω, Big-Θ notations.
 - **Key Concepts:** Worst, Best, and Average Case Time; Space complexity of recursive & iterative code.
 - **Reference Material:**
 - Read: <https://www.geeksforgeeks.org/analysis-of-algorithms-set-1-asymptotic-analysis/>
 - **Practice (Understanding Only):**
 - <https://leetcode.com/discuss/general-discussion/1127238/time-complexity-of-leetcode-problems>
-

Day 2: Arrays - Basics

- **Goal:** Learn array declaration, traversal, and simple problems.
 - **Problems:**
 - Two Sum
<https://leetcode.com/problems/two-sum/>
 - Best Time to Buy and Sell Stock
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>
 - **Concepts Covered:** Brute-force vs Hashmap approach, Greedy logic.
-

Day 3: Arrays - Sliding Window

- **Goal:** Master fixed and variable size sliding window.
 - **Problems:**
 - Maximum Subarray (Kadane's Algo)
<https://leetcode.com/problems/maximum-subarray/>
 - Longest Substring Without Repeating Characters
<https://leetcode.com/problems/longest-substring-without-repeating-characters/>
 - **Concepts Covered:** Optimizing brute-force with window compression.
-

Day 4: Arrays - Prefix Sum

- **Goal:** Learn cumulative sum techniques for range queries.
 - **Problems:**
 - Range Sum Query - Immutable
<https://leetcode.com/problems/range-sum-query-immutable/>
 - Subarray Sum Equals K
<https://leetcode.com/problems/subarray-sum-equals-k/>
 - **Concepts Covered:** Storing previous sums, Hashmap optimization.
-

Day 5: Arrays - Binary Search

- **Goal:** Practice standard binary search and modifications.
 - **Problems:**
 - Binary Search
<https://leetcode.com/problems/binary-search/>
 - Find First and Last Position of Element in Sorted Array
<https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/>
 - **Concepts Covered:** Left/Right boundary techniques, Log(N) time.
-

Day 6: Arrays - Two Pointers

- **Goal:** Learn how to approach problems with two pointers from start/end.
 - **Problems:**
 - Container With Most Water
<https://leetcode.com/problems/container-with-most-water/>
 - 3Sum
<https://leetcode.com/problems/3sum/>
 - **Concepts Covered:** Sorting + two pointers, avoiding duplicates.
-

Day 7: Arrays - Sorting Techniques

- **Goal:** Practice sorting-based approaches.
- **Problems:**
 - Merge Intervals
<https://leetcode.com/problems/merge-intervals/>
 - Sort Colors (Dutch National Flag)
<https://leetcode.com/problems/sort-colors/>
- **Concepts Covered:** Sorting arrays with logic, not just .sort().

Week 2: Strings + Hashing

Day 8: Strings - Basics & Character Count

- **Goal:** Understand basic string manipulations.
 - **Problems:**
 - Valid Anagram
<https://leetcode.com/problems/valid-anagram/>
 - Ransom Note
<https://leetcode.com/problems/ransom-note/>
 - **Concepts Covered:** Hashmaps, character frequency, ord(), Counter() in Python.
-

Day 9: Strings - Palindrome Problems

- **Goal:** Master checking palindromes and variations of it.
 - **Problems:**
 - Valid Palindrome
<https://leetcode.com/problems/valid-palindrome/>
 - Longest Palindromic Substring
<https://leetcode.com/problems/longest-palindromic-substring/>
 - **Concepts Covered:** Two pointers, expanding around center, ignoring non-alphanumeric characters.
-

Day 10: Strings - Two Pointer + Hashing

- **Goal:** Combine two techniques for efficient string problems.
 - **Problems:**
 - Reverse Vowels of a String
<https://leetcode.com/problems/reverse-vowels-of-a-string/>
 - Isomorphic Strings
<https://leetcode.com/problems/isomorphic-strings/>
 - **Concepts Covered:** Mapping between two strings, positional character hashing.
-

Day 11: Strings - Sliding Window on Strings

- **Goal:** Apply sliding window technique to strings.
- **Problems:**
 - Minimum Window Substring
<https://leetcode.com/problems/minimum-window-substring/>
 - Permutation in String
<https://leetcode.com/problems/permutation-in-string/>

- **Concepts Covered:** Dynamic sliding window, character window frequency tracking.
-

Day 12: HashMap - Fundamentals

- **Goal:** Practice HashMap operations and applications.
 - **Problems:**
 - Two Sum (again, but for hashing logic)
<https://leetcode.com/problems/two-sum/>
 - Group Anagrams
<https://leetcode.com/problems/group-anagrams/>
 - **Concepts Covered:** Hashmaps for grouping data, using tuples as dictionary keys.
-

Day 13: HashSet + Duplicates

- **Goal:** Learn to handle duplicate detection using sets.
 - **Problems:**
 - Contains Duplicate
<https://leetcode.com/problems/contains-duplicate/>
 - Longest Consecutive Sequence
<https://leetcode.com/problems/longest-consecutive-sequence/>
 - **Concepts Covered:** Set operations, O(n) optimizations.
-

Day 14: Wrap-up & Revision

- **Tasks:**
 - Revisit any problems you found difficult.
 - Re-attempt at least 3 previously solved problems without looking.
 - Watch a revision video on Strings & Hashing (e.g., from NeetCode or Take U Forward).
- **Bonus Challenge:** Try at least one medium-hard problem like:
 - Encode and Decode Strings
<https://leetcode.com/problems/encode-and-decode-strings/>

Week 3: Linked Lists

Day 15: Linked List Basics

- **Goal:** Understand singly linked list fundamentals.
 - **Problems:**
 - Reverse Linked List
<https://leetcode.com/problems/reverse-linked-list/>
 - Middle of the Linked List
<https://leetcode.com/problems/middle-of-the-linked-list/>
 - **Concepts Covered:** Fast and slow pointers, reversing using pointers.
-

Day 16: Linked List Insertion and Deletion

- **Goal:** Master node manipulation techniques.
 - **Problems:**
 - Delete Node in a Linked List
<https://leetcode.com/problems/delete-node-in-a-linked-list/>
 - Remove Nth Node From End of List
<https://leetcode.com/problems/remove-nth-node-from-end-of-list/>
 - **Concepts Covered:** Pointer shifting, one-pass deletion using fast and slow pointer.
-

Day 17: Detecting Cycles in Linked List

- **Goal:** Detect and handle cycles.
 - **Problems:**
 - Linked List Cycle
<https://leetcode.com/problems/linked-list-cycle/>
 - Linked List Cycle II
<https://leetcode.com/problems/linked-list-cycle-ii/>
 - **Concepts Covered:** Floyd's Cycle Detection Algorithm (Tortoise and Hare).
-

Day 18: Merge & Sort Linked Lists

- **Goal:** Work with sorting and merging operations.
- **Problems:**
 - Merge Two Sorted Lists
<https://leetcode.com/problems/merge-two-sorted-lists/>
 - Sort List
<https://leetcode.com/problems/sort-list/>
- **Concepts Covered:** Merge sort on linked lists, recursion.

Day 19: Advanced Reversals

- **Goal:** Reverse parts of a linked list.
 - **Problems:**
 - Reverse Linked List II
<https://leetcode.com/problems/reverse-linked-list-ii/>
 - Reverse Nodes in k-Group
<https://leetcode.com/problems/reverse-nodes-in-k-group/>
 - **Concepts Covered:** Pointer juggling, group reversal, dummy nodes.
-

Day 20: Intersection & Palindrome

- **Goal:** Learn to check for intersection and palindromes.
 - **Problems:**
 - Intersection of Two Linked Lists
<https://leetcode.com/problems/intersection-of-two-linked-lists/>
 - Palindrome Linked List
<https://leetcode.com/problems/palindrome-linked-list/>
 - **Concepts Covered:** Stack-based and pointer-based approaches, length alignment.
-

Day 21: Wrap-up & Revision

- **Tasks:**
 - Re-solve 2-3 tricky problems from earlier in the week.
 - Draw diagrams for Linked List Cycle and Reverse Nodes in K Groups.
 - Optional practice problem:
 - Add Two Numbers
<https://leetcode.com/problems/add-two-numbers/>

Week 4: Stacks and Queues

Day 22: Stack Basics

- **Goal:** Understand the fundamentals of stack operations (LIFO).
 - **Problems:**
 - Valid Parentheses
<https://leetcode.com/problems/valid-parentheses/>
👉 Use stack to match opening and closing brackets.
 - Min Stack
<https://leetcode.com/problems/min-stack/>
👉 Maintain a second stack to keep track of minimum values.
-

Day 23: Stack Advanced Applications

- **Goal:** Handle stack problems involving indices and nested structures.

- **Problems:**

- Next Greater Element I
<https://leetcode.com/problems/next-greater-element-i/>
👉 Use stack in reverse traversal for efficient solution.
 - Next Greater Element II
<https://leetcode.com/problems/next-greater-element-ii/>
👉 Handle circular array using modulo and double loop logic.
-

Day 24: Monotonic Stack

- **Goal:** Learn the concept of increasing/decreasing stack.
 - **Problems:**
 - Daily Temperatures
<https://leetcode.com/problems/daily-temperatures/>
👉 Store indices to calculate number of days until warmer temp.
 - Asteroid Collision
<https://leetcode.com/problems/asteroid-collision/>
👉 Use stack to simulate collisions based on sizes and directions.
-

Day 25: Queue Basics

- **Goal:** Learn FIFO structure and implement using arrays or lists.
 - **Problems:**
 - Implement Queue using Stacks
<https://leetcode.com/problems/implement-queue-using-stacks/>
👉 Use two stacks to simulate queue behavior.
 - Number of Recent Calls
<https://leetcode.com/problems/number-of-recent-calls/>
👉 Use queue to count requests in a rolling time window.
-

Day 26: Deque (Double-ended Queue)

- **Goal:** Learn how to insert and delete from both ends efficiently.
 - **Problems:**
 - Sliding Window Maximum
<https://leetcode.com/problems/sliding-window-maximum/>
👉 Use deque to maintain decreasing order of elements for O(n) solution.
 - Design Circular Deque
<https://leetcode.com/problems/design-circular-deque/>
👉 Implement custom deque with capacity handling.
-

Day 27: Stack-Queue Hybrid Problems

- **Goal:** Solve real-world problems involving both data structures.
 - **Problems:**
 - Evaluate Reverse Polish Notation
<https://leetcode.com/problems/evaluate-reverse-polish-notation/>
👉 Use stack to simulate RPN evaluation.
 - Implement Stack using Queues
<https://leetcode.com/problems/implement-stack-using-queues/>
👉 Queue-based simulation of LIFO operations.
-

Day 28: Wrap-up & Revision

- **Tasks:**

- Re-attempt tricky problems like Daily Temperatures or Sliding Window Maximum.
- Practice visualization and dry runs.
- Optional Challenge:
 - Decode String
<https://leetcode.com/problems/decode-string/>
👉 Stack-based problem for nested encoding like "3[a2[c]]" → "accaccacc"

Week 5: Recursion & Backtracking

Day 29: Basics of Recursion

- **Goal:** Understand the concept of recursion (base + recursive case).
 - **Problems:**
 - Factorial of a Number (custom implementation)  Write your own recursive code to understand the call stack.
 - Fibonacci Number
<https://leetcode.com/problems/fibonacci-number/>
👉 Recursively compute Fibonacci; later memoize for optimization.
-

Day 30: Recursion Problems

- **Goal:** Apply recursion to solve small problems.
 - **Problems:**
 - Reverse Linked List (Recursive)
<https://leetcode.com/problems/reverse-linked-list/>
👉 Use recursive calls to reverse nodes one-by-one.
 - Power of Three
<https://leetcode.com/problems/power-of-three/>
👉 Recursively divide the number by 3 to check base case.
-

Day 31: Backtracking Intro

- **Goal:** Learn how to explore all possible solutions via recursion + undoing choices.
 - **Problems:**
 - Subsets
<https://leetcode.com/problems/subsets/>
👉 Use recursion to generate all possible combinations (include/exclude).
 - Combination Sum
<https://leetcode.com/problems/combination-sum/>
👉 Try all candidates recursively and backtrack when sum exceeds target.
-

Day 32: Backtracking - Combinations & Permutations

- **Goal:** Understand permutation logic and constraints.
 - **Problems:**
 - Permutations
<https://leetcode.com/problems/permutations/>
👉 Swap-based or used-array-based recursive solution.
 - Combinations
<https://leetcode.com/problems/combinations/>
👉 Standard backtracking, maintain start index for no repeats.
-

Day 33: Backtracking - Include/Exclude Pattern

- **Goal:** Learn subset and decision-tree style problems.
 - **Problems:**
 - Subsets II (handle duplicates)
<https://leetcode.com/problems/subsets-ii/>
👉 Sort array and skip duplicates in recursion.
 - Letter Combinations of a Phone Number
<https://leetcode.com/problems/letter-combinations-of-a-phone-number/>
👉 Map digits to letters and backtrack all combinations.
-

Day 34: Backtracking - Constraints

- **Goal:** Understand how to deal with constraint-based paths.
 - **Problems:**
 - N-Queens
<https://leetcode.com/problems/n-queens/>
👉 Place queens row by row and backtrack when there's a clash.
 - Word Search
<https://leetcode.com/problems/word-search/>
👉 Recursively explore adjacent cells and backtrack.
-

Day 35: Recap + Challenge

- **Goal:** Solidify concepts and push your limits.
- **Tasks:**
 - Re-attempt any tricky backtracking problems.
 - Optional Challenges:
 - Sudoku Solver
<https://leetcode.com/problems/sudoku-solver/>
👉 Recursive DFS with validation and backtracking.
 - Generate Parentheses
<https://leetcode.com/problems/generate-parentheses/>
👉 Backtrack with open/close balance logic.

Week 6: Binary Trees and BSTs

Day 36: Binary Tree Basics

- **Goal:** Understand tree structure and traversal basics.
- **Problems:**
 - Invert Binary Tree
<https://leetcode.com/problems/invert-binary-tree/>
👉 Use recursion to swap left and right subtrees.
 - Maximum Depth of Binary Tree
<https://leetcode.com/problems/maximum-depth-of-binary-tree/>
👉 Return 1 + max depth of left and right recursively.

Day 37: Tree Traversals

- **Goal:** Practice in-order, pre-order, and post-order traversals.
 - **Problems:**
 - Binary Tree Inorder Traversal
<https://leetcode.com/problems/binary-tree-inorder-traversal/>
👉 Practice both recursive and iterative approaches.
 - Binary Tree Preorder Traversal
<https://leetcode.com/problems/binary-tree-preorder-traversal/>
👉 Similar to inorder but root is visited first.
-

Day 38: Level Order and Zigzag

- **Goal:** Understand BFS on trees.
 - **Problems:**
 - Binary Tree Level Order Traversal
<https://leetcode.com/problems/binary-tree-level-order-traversal/>
👉 Use queue to process level by level.
 - Binary Tree Zigzag Level Order Traversal
<https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/>
👉 Flip direction on each level using deque.
-

Day 39: Lowest Common Ancestor (LCA)

- **Goal:** Learn about tree ancestors and recursive backtracking.
 - **Problems:**
 - Lowest Common Ancestor of a Binary Tree
<https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/>
👉 Recurse from root to find paths to both nodes.
 - Path Sum
<https://leetcode.com/problems/path-sum/>
👉 Check if a root-to-leaf path sums up to target.
-

Day 40: Diameter & Balanced Tree

- **Goal:** Understand tree properties (height, balance).
- **Problems:**
 - Diameter of Binary Tree
<https://leetcode.com/problems/diameter-of-binary-tree/>
👉 Max of left + right heights at each node.

- Balanced Binary Tree
<https://leetcode.com/problems/balanced-binary-tree/>
👉 Tree is balanced if left and right subtrees differ by ≤ 1 .
-

Day 41: BST - Basics

- **Goal:** Practice BST-specific operations.
 - **Problems:**
 - Validate Binary Search Tree
<https://leetcode.com/problems/validate-binary-search-tree/>
👉 Check node values within min-max bounds.
 - Search in a Binary Search Tree
<https://leetcode.com/problems/search-in-a-binary-search-tree/>
👉 Standard BST search - go left or right.
-

Day 42: BST - Insert & Delete

- **Goal:** Understand structural modifications in BST.
 - **Problems:**
 - Insert into a Binary Search Tree
<https://leetcode.com/problems/insert-into-a-binary-search-tree/>
👉 Recursively insert at the right position.
 - Delete Node in a BST
<https://leetcode.com/problems/delete-node-in-a-bst/>
👉 Handle cases for leaf, one child, or two children.
-

Tips for Week 6:

- Use recursion to your advantage in trees.
- Dry run examples on paper.
- Don't memorize – **understand the tree structure and traversal patterns.**

✓ Week 7: Heaps, Tries & Graphs (Intro)

💻 Day 43: Heaps Basics

- **Goal:** Understand Min-Heap and Max-Heap concepts and priority queue.
- **Problems:**
 - Kth Largest Element in an Array
<https://leetcode.com/problems/kth-largest-element-in-an-array/>
👉 Use a **Min-Heap of size k** to keep track of k largest elements.
 - Top K Frequent Elements
<https://leetcode.com/problems/top-k-frequent-elements/>
👉 Build frequency map, use **heapq.nlargest** with a custom key.

💻 Day 44: Sliding Window + Heaps

- **Goal:** Practice heap use with sliding window technique.
- **Problems:**
 - Sliding Window Maximum
<https://leetcode.com/problems/sliding-window-maximum/>
👉 Use deque or heap with (value, index) pairs, pop out-of-window items.
 - Find Median from Data Stream
<https://leetcode.com/problems/find-median-from-data-stream/>
👉 Use two heaps: max-heap for left, min-heap for right half.

💻 Day 45: Introduction to Tries

- **Goal:** Understand Trie data structure and implementation.
- **Problems:**
 - Implement Trie (Prefix Tree)
<https://leetcode.com/problems/implement-trie-prefix-tree/>
👉 Classic Trie implementation with insert, search, and startsWith.
 - Replace Words
<https://leetcode.com/problems/replace-words/>
👉 Build a Trie of root words, replace sentence words with the root if matched.

Day 46: Word Problems Using Trie

- **Goal:** Practice string manipulation with Trie optimization.
 - **Problems:**
 - Word Search II
<https://leetcode.com/problems/word-search-ii/>
👉 Combine Trie + DFS to efficiently search multiple words.
 - Design Add and Search Words Data Structure
<https://leetcode.com/problems/design-add-and-search-words-data-structure/>
👉 Trie with support for dot . wildcard search.
-

Day 47: Graph Basics (Adjacency List, DFS)

- **Goal:** Get comfortable with graph representation & DFS.
 - **Problems:**
 - Number of Islands
<https://leetcode.com/problems/number-of-islands/>
👉 Use DFS to mark visited nodes in grid.
 - Flood Fill
<https://leetcode.com/problems/flood-fill/>
👉 Classic DFS/BFS to fill connected region with new color.
-

Day 48: Graph BFS & DFS Practice

- **Goal:** Explore both DFS and BFS patterns in real scenarios.
 - **Problems:**
 - Clone Graph
<https://leetcode.com/problems/clone-graph/>
👉 DFS or BFS with hashmap to track cloned nodes.
 - Rotten Oranges
<https://leetcode.com/problems/rotting-oranges/>
👉 BFS with queue for time-based state changes.
-

Day 49: Cycle Detection

- **Goal:** Learn how to detect cycles in graphs.
- **Problems:**
 - Course Schedule
<https://leetcode.com/problems/course-schedule/>
👉 Use DFS with visited and recursion stack.

- Detect Cycle in Directed Graph (GFG)
<https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1>
👉 Classic DFS cycle detection.
-

Tips for Week 7:

- Practice heap operations with Python's heapq (default is min-heap).
 - For Trie: Create a TrieNode class with children = {} and end = False.
 - Understand graph traversal clearly – try drawing graphs on paper.
-

Week 8: Graphs (Advanced)

Day 50: Topological Sort (DFS & BFS/Kahn's Algo)

- **Goal:** Learn how to order nodes in a Directed Acyclic Graph (DAG).
 - **Problems:**
 - Course Schedule II
<https://leetcode.com/problems/course-schedule-ii/>
👉 Solve using Kahn's Algorithm (BFS) or DFS + stack.
 - Topological Sort (GFG)
<https://practice.geeksforgeeks.org/problems/topological-sort/1>
👉 Practice both DFS and Kahn's approach.
-

Day 51: Shortest Path in Graphs

- **Goal:** Understand Dijkstra's Algorithm & BFS for unweighted graphs.
 - **Problems:**
 - Network Delay Time
<https://leetcode.com/problems/network-delay-time/>
👉 Use Dijkstra with min-heap for shortest path.
 - Shortest Path in Binary Matrix
<https://leetcode.com/problems/shortest-path-in-binary-matrix/>
👉 Use BFS for unweighted grid.
-

Day 52: Bellman-Ford and Negative Cycles

- **Goal:** Handle negative weights using Bellman-Ford.
 - **Problems:**
 - Bellman Ford (GFG)
<https://practice.geeksforgeeks.org/problems/distance-from-the-source-bellman-ford-algorithm/1>
👉 Relax all edges V-1 times.
 - Negative Weight Cycle (GFG)
<https://practice.geeksforgeeks.org/problems/negative-weight-cycle3504/1>
👉 After V-1 iterations, check for further relaxation (i.e., cycle).
-

Day 53: Disjoint Set (Union Find) - Part 1

- **Goal:** Understand Union by Rank & Path Compression.
 - **Problems:**
 - Number of Connected Components in an Undirected Graph
<https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph/>
👉 Count unique parents in Union-Find structure.
 - Disjoint Set Union (GFG)
<https://practice.geeksforgeeks.org/problems/disjoint-set-union-find/1>
👉 Classic union-find with optimizations.
-

Day 54: Disjoint Set (Union Find) - Part 2

- **Goal:** Practice harder variations.
 - **Problems:**
 - Redundant Connection
<https://leetcode.com/problems/redundant-connection/>
👉 Add edges and detect cycle using DSU.
 - Accounts Merge
<https://leetcode.com/problems/accounts-merge/>
👉 Map emails to DSU, merge accounts.
-

Day 55: Minimum Spanning Tree (MST)

- **Goal:** Learn Kruskal's and Prim's algorithms.
 - **Problems:**
 - Connecting Cities With Minimum Cost (GFG)
<https://practice.geeksforgeeks.org/problems/connecting-the-graph/1>
👉 Kruskal's Algorithm with DSU.
 - Minimum Cost to Connect All Points
<https://leetcode.com/problems/min-cost-to-connect-all-points/>
👉 Use Prim's or Kruskal's with heap/graph.
-

Day 56: Graph Recap + Practice

- **Goal:** Review concepts and revisit weak areas.
 - **Suggested Practice:**
 - Practice 3 random graph problems from LeetCode's Graph tag:
<https://leetcode.com/tag/graph/>
 - Graph Topic Revision Sheet (Love Babbar / Striver)
<https://takeuforward.org/graph/>
-

Tips for Week 8:

-  Remember: BFS = queue, DFS = recursion/stack.
-  Graphs are used in maps, networks, scheduling, web crawlers.
-  Disjoint Sets are  in cycle detection, Kruskal's MST, and network problems.
-  Don't rush – understanding > speed.

Week 9: Dynamic Programming I – Basics, 0/1 Knapsack, LIS, Memoization, and Tabulation

Dynamic Programming is the soul of interviews. It's like cooking: prep your base (recursion), spice it up (memoization), and plate it like a pro (tabulation) 

Day 57: Intro to DP + Fibonacci + Climbing Stairs

- **Goal:** Understand the recursion-to-DP transition.
 - **Problems:**
 - Fibonacci Number
<https://leetcode.com/problems/fibonacci-number/>
➤ Try all 3 methods: Recursion → Memoization → Tabulation.
 - Climbing Stairs
<https://leetcode.com/problems/climbing-stairs/>
➤ Classic DP, like Fibonacci – start from the end.
-

Day 58: 0/1 Knapsack - Recursion + Memoization

- **Goal:** Learn base DP: choose/don't choose item.
 - **Problems:**
 - 0-1 Knapsack Problem (GFG)
<https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem0945/1>
➤ Apply recursion → memoization. Core concept of DP.
 - Subset Sum Problem
<https://practice.geeksforgeeks.org/problems/subset-sum-problem2014/1>
➤ Base for knapsack. Try memoized recursion.
-

Day 59: 0/1 Knapsack - Tabulation + Space Optimization

- **Goal:** Convert memoized version into bottom-up DP.
 - **Problems:**
 - Equal Sum Partition
<https://practice.geeksforgeeks.org/problems/partition-equal-subset-sum/0>
➤ Convert subset sum to tabulation.
 - Count of Subset Sum (GFG)
<https://practice.geeksforgeeks.org/problems/perfect-sum-problem5633/1>
➤ Try using 1D space optimization after tabulation.
-

Day 60: Longest Common Subsequence (LCS)

- **Goal:** Intro to 2D DP problems.
 - **Problems:**
 - Longest Common Subsequence
<https://leetcode.com/problems/longest-common-subsequence/>
► Solve via memoization & tabulation.
 - Print LCS (GFG)
<https://www.geeksforgeeks.org/print-longest-common-subsequence/>
► Modify tabulation table to backtrack and print sequence.
-

Day 61: LCS Variants

- **Goal:** Tweak LCS logic for variations.
 - **Problems:**
 - Longest Palindromic Subsequence
<https://leetcode.com/problems/longest-palindromic-subsequence/>
► Use LCS of string and its reverse.
 - Shortest Common Supersequence
<https://leetcode.com/problems/shortest-common-supersequence/>
► SCS = (m + n) - LCS.
-

Day 62: Longest Increasing Subsequence (LIS)

- **Goal:** Classic 1D DP with $O(n^2)$ and $O(n \log n)$ solutions.
 - **Problems:**
 - Longest Increasing Subsequence
<https://leetcode.com/problems/longest-increasing-subsequence/>
► Solve with DP (n^2), then try binary search method.
 - Russian Doll Envelopes
<https://leetcode.com/problems/russian-doll-envelopes/>
► Convert to LIS after sorting envelopes.
-

Day 63: DP Recap + Practice

- **Goal:** Practice all types: 1D, 2D, knapsack, LCS, LIS.
 - **Suggestions:**
 - DP Playlist by Striver:
<https://www.youtube.com/playlist?list=PLgUwDviBIf0rGEWe64KWas0Nryn7SCRWw>
 - Random practice from LeetCode DP tag:
<https://leetcode.com/tag/dynamic-programming/>
-

Summary of Patterns This Week:

Pattern	Real-life Analogy
Climbing Stairs	Ways to climb a ladder 
Knapsack	Packing a bag with max value 
LCS / LPS	Text diff or DNA sequencing 
LIS	Longest growth trend 

Week 10: Advanced DP – Grids, Trees, MCM, Partitions

Day 64: DP on Grids – Basics

- **Goal:** Learn to move through grids using DP.
 - **Problems:**
 - Unique Paths
<https://leetcode.com/problems/unique-paths/>
► Grid DP – try recursion, memoization, then tabulation.
 - Minimum Path Sum
<https://leetcode.com/problems/minimum-path-sum/>
► Use the same approach – reuse grid states.
-

Day 65: DP on Grids – Obstacles

- **Goal:** Add edge cases to grid DP.
- **Problems:**
 - Unique Paths II (Obstacles)
<https://leetcode.com/problems/unique-paths-ii/>
► If $\text{grid}[i][j] == 1$, block the path.

- Cherry Pickup II
<https://leetcode.com/problems/cherry-pickup-ii/>
► Hard level grid DP with 2 robots → 3D DP.
-

Day 66: DP on Trees – Basics

- **Goal:** Solve problems with recursive DFS + memo.
 - **Problems:**
 - House Robber III
<https://leetcode.com/problems/house-robber-iii/>
► Apply DP with postorder DFS on tree.
 - Diameter of Binary Tree
<https://leetcode.com/problems/diameter-of-binary-tree/>
► Use DP logic on left and right subtrees.
-

Day 67: DP on Trees – Practice

- **Goal:** Master tree-based recursion.
 - **Problems:**
 - Binary Tree Cameras
<https://leetcode.com/problems/binary-tree-cameras/>
► Postorder DP with three states.
 - Longest ZigZag Path in a Binary Tree
<https://leetcode.com/problems/longest-zigzag-path-in-a-binary-tree/>
► Alternate directions while traversing.
-

Day 68: MCM – Matrix Chain Multiplication Intro

- **Goal:** Understand the MCM base pattern.
 - **Problems:**
 - Matrix Chain Multiplication (GFG)
<https://www.geeksforgeeks.org/matrix-chain-multiplication-dp-8/>
► Try recursion → memoization.
 - Burst Balloons
<https://leetcode.com/problems/burst-balloons/>
► Convert to MCM form – like cutting balloons at i.
-

Day 69: MCM Variants

- **Goal:** Apply MCM pattern to other partition problems.
- **Problems:**
 - Palindrome Partitioning II

<https://leetcode.com/problems/palindrome-partitioning-ii/>

► Min number of cuts → Try recursion + memo.

- Boolean Parenthesization (GFG)

<https://www.geeksforgeeks.org/boolean-parenthesization-problem-dp-37/>

► Classic MCM + multiple return values.

Day 70: DP Grand Finale Practice

- **Goal:** Revise grids + trees + MCM together!

- **Suggestions:**

- DP on Trees GFG Set:

<https://practice.geeksforgeeks.org/explore?page=1&category%5B%5D=Tree&sortBy=submissions>

- Random LeetCode DP tag practice:

<https://leetcode.com/tag/dynamic-programming/>

Summary of DP Types:

Type	Keyword	Analogy
Grid DP	"Move through matrix"	Robot walking in a maze 
Tree DP	"Postorder with memo"	Robbing house in a colony 
MCM	"i to j partition"	Breaking problems recursively 

Daily Schedule Template (2.5 Hours / Day)

Morning (Optional Warm-up - 15 min)

- **Activity:** Watch a short video / revisit notes from the previous day.
 - **Goal:** Refresh concepts.
-

Evening / Main Session (2 Hours)

Time Slot	Activity	Details
0:00 - 0:20	Concept Study	Read/Watch theory for the day's topic (e.g., Recursion, Trees)
0:20 - 1:30	Problem Solving	Solve 2 core problems from LeetCode or GFG (links from roadmap)
1:30 - 2:00	Discuss / Revise	Revisit tricky logic, write comments, compare with discussions

Night (Optional - 15 min)

- **Activity:** Update your **Study Tracker** or **Anki Flashcards**.
- **Goal:** Reinforce logic and patterns.

Weekend Plan (Every 7th Day)

Time	Activity
1 Hour	Revise Week's Topics (watch summary / revisit notes)
1 Hour	Re-solve 2 Tough Problems
30 mins	Mock Test / Timed Contest (LeetCode Weekly / Biweekly)
15 mins	Update Progress Tracker

Tips to Stay Consistent:

-  Use a printed tracker or Notion board – tick off problems daily.
-  Don't rush hard concepts (like DP or Graphs). Split over 2 days if needed.
-  Discuss tough problems with peers / Discord / Reddit / LeetCode Discuss.
-  Reward yourself weekly for completion (chill time, fav snack, etc.).

Most Frequently Asked Interview Questions

1. Arrays & Strings

- **Two Sum**
Find indices of two numbers that add up to a target.
- **Best Time to Buy and Sell Stock**
Maximize profit by choosing buy/sell days.
- **Container With Most Water**
Given heights array, find max “water” between two lines.
- **Longest Substring Without Repeating Characters**
Sliding-window to find max-length substring with unique chars.
- **3Sum**
Find all unique triplets that sum to zero.
- **Product of Array Except Self**
Compute products without using division in O(n).

2. Linked Lists

- **Reverse a Linked List**
Iterative and recursive approaches.
- **Detect Cycle in Linked List**
Floyd's Tortoise and Hare algorithm.

- **Merge Two Sorted Lists**
Merge in-place or by creating a new list.
- **Add Two Numbers**
Sum two numbers represented by reversed linked lists.
- **Remove Nth Node From End**
Two-pointer technique for one-pass removal.

2. Stacks & Queues

- **Valid Parentheses**
Use stack to check matching pairs in a string.
- **Min Stack**
Design a stack that returns the minimum element in O(1).
- **Number of Islands** (also Graph/DFS)
Count connected components in a 2D grid.

3. Trees & Graphs

- **Binary Tree Inorder/Preorder/Postorder Traversal**
Recursive and iterative with stack.
- **Lowest Common Ancestor of a BST/BT**
Use tree properties or parent pointers.
- **Serialize and Deserialize Binary Tree**
Encode tree structure into string and rebuild.
- **Number of Islands**
(grid as implicit graph) flood-fill / DFS / BFS.
- **Course Schedule**
Detect cycles in a directed graph (topological sort).

4. Heaps & Hashing

- **Top K Frequent Elements**
Use heap or bucket-sort on frequencies.
- **Kth Largest Element in an Array**
Quickselect or min-heap of size k.
- **LRU Cache**
Design with O(1) get/put using hash + doubly-linked list.
- **Rain Water Trapping**
Compute trapped water using two-pointer or stack.

5. Dynamic Programming

- **Climbing Stairs / Fibonacci Variants**
Simple DP for linear recurrences.
- **House Robber**
Max sum without adjacent selections.

- **Longest Increasing Subsequence**
Patience sorting ($O(n \log n)$) or DP ($O(n^2)$).
- **Coin Change**
Min coins to make amount.
- **Edit Distance**
Levenshtein distance via 2D DP.

6. Sorting & Searching

- **Merge Intervals**
Sort by start, then merge overlaps.
- **Search in Rotated Sorted Array**
Modified binary search.
- **Median of Two Sorted Arrays**
Divide-and-conquer in $O(\log (\min m, n))$.

7. Advanced Topics

- **Trie (Prefix Tree) Implementation**
Insert/search/delete words.
- **Segment Tree / Fenwick Tree**
Range-sum and point-update queries.
- **Sliding Window Maximum**
Deque-based $O(n)$ solution.
- **Word Ladder**
BFS on word-transform graph.
- **Alien Dictionary**
Topological sort on custom alphabet.

Tips for practice:

1. **Master patterns** (two pointers, sliding window, divide & conquer, greedy, backtracking, graph traversals, DP).
2. **Start with easy-level**, then medium, then hard.
3. **Write clean, tested code**—include edge cases.
4. **Analyze time/space complexity** for each solution.