DSA Roadmap

Phase 1: Foundational Structures & Analysis (Weeks 1-5)

Week	**Focus Area**	**Key Concepts**	**Recommended Resources (Free/Low-Cost)**	**Practice Goal (Platform)**
1	**Analysis & Arrays**	**Big O Notation** (Time/Space Complexity), Trade-offs, Arrays (Fixed/Dynamic), Pointers.	**HackerRank** (Algorithms - Intro), **GeeksforGeeks** (Big O), NeetCode (Basics).	Solve 15+ easy Array problems (e.g., Two Sum, finding min/max).
2	**Linked Lists**	Singly, Doubly, and Circular Linked Lists, Insertion/Deletion, Sentinel Nodes.	**LeetCode** (Linked List - Easy/Medium), **Visualizing Algorithms** tools.	Solve 10+ problems (e.g., Reverse a Linked List, detect a cycle).
3	**Stacks & Queues**	LIFO/FIFO principles, implementation using Arrays/Lists, common applications (DFS, BFS precursors).	Tutorials on Infix/Postfix conversion, **LeetCode/HackerRank** (Stack/Queue).	Implement **Infix to Postfix** conversion; implement a Queue using two Stacks.

4	**Hash Maps & Sets**	**Hashing** principles, collision resolution, Time complexity of lookups, Trade-offs vs. Trees.	**NeetCode** (Arrays & Hashing), **GeeksforGeeks** (Hashing).	Solve 15+ problems utilizing hash maps for fast lookups (e.g., Group Anagrams).
5	**Trees**	Binary Trees (BT), Binary Search Trees (**BST**), Traversal methods (**BFS, DFS** - Inorder/Preorder/Postorder).	**freeCodeCamp** (DSA course), **LeetCode** (Tree Traversal).	Implement all three DFS traversals; solve problems involving BST validation.

Phase 2: Algorithms & Advanced Structures (Weeks 6-10)

Week	**Focus Area**	**Key Concepts**	**Recommended Resources (Free/Low-Cost)**	**Practice Goal (Platform)**
6	**Heaps & Priority Queues**	Min-Heap/Max-Heap implementation, Heapify process, Priority Queue applications.	**GeeksforGeeks** (Heap), Tutorials on K-th largest element problems.	Solve 5+ problems involving finding the K-th smallest/largest element.

7	**Sorting & Searching**	**Binary Search** (Iterative/Recursive), Merge Sort, Quick Sort, Comparison of time complexities ($O(n \log n)$ vs. $O(n^2)$).	**Grokking Algorithms** (Book/Concepts), Practice Binary Search edge cases.	Implement Merge Sort and Quick Sort from scratch.
8	**Graphs (Part 1)**	Graph representations (Adjacency List/Matrix), **Breadth-First Search (BFS)**, **Depth-First Search (DFS)**, connectivity.	**Aditya Verma** (YouTube - Graphs), **LeetCode** (Graph Traversal).	Implement BFS/DFS; solve problems involving connected components.
9	**Graphs (Part 2)**	Topological Sort, Dijkstra's Algorithm (Shortest Path), Minimum Spanning Tree (Prim's/Kruskal's conceptual).	**MIT OpenCourseWare** (Algorithms), Tutorials on graph-specific algorithms.	Solve a problem requiring **Topological Sort** (e.g., Course Schedule).
10	**Greedy Algorithms**	Principle of making locally optimal choices, Identifying when a greedy approach works (and when it fails).	Examples like Activity Selection Problem, Fractional Knapsack.	Solve 5-7 medium-level problems using a **Greedy strategy**.

Phase 3: Dynamic Programming & Interview Skills (Weeks 11-14)

Week	**Focus Area**	**Key Concepts**	**Recommended Resources (Free/Low-Cost)**	**Project / Practice Goal**

11	**Recursion & Backtracking**	Base cases, Recursive calls, Call Stack tracing, Applications in Sudoku Solver, Permutations/Combinations.	**Backtracking visualization** tools, **HackerRank** (Recursion).	Implement solutions for Permutations and Combinations problems.
12	**Dynamic Programming (DP)**	**Memoization** (Top-Down), **Tabulation** (Bottom-Up), Recognizing optimal substructure and overlapping subproblems.	**Aditya Verma** (DP Playlist), **Grokking Dynamic Programming** patterns.	Solve classic DP problems (e.g., Fibonacci, Coin Change, Knapsack).
13	**Behavioral & System Design**	**STAR Method** for behavioral questions, Handling failure/conflict, Introduction to **System Design** (Conceptual: Load Balancers, Databases).	**UpStride Mentor** sessions, **Grokking System Design** (Conceptual).	Write out 5 detailed STAR-method answers; discuss the design of a popular service (e.g., Netflix).
14	**Final Review & Mock Interviews**	Review of **"Blind 75"** or similar top interview question lists, Final mock interviews, **Personalized feedback**.	**LeetCode** (Top Interview Questions List), **UpStride Mock Interview Platform**.	Complete 3 focused mock interviews with detailed feedback and practice time constraint.

What's Next? Your Path After 14 Weeks

Career Action	**Specific Goal / Outcome**	**Tools & Resources**
Deepen Portfolio	**Maintain Daily Practice:** Practice 1-2 medium/hard DSA problems daily to keep the muscle strong. Focus on System Design concepts.	**LeetCode (Daily Challenge)**, **Interviewing.io** (System Design topics), **Grokking System Design** (advanced topics).
Specialized Skills	Apply DSA to a specialized area: **Competitive Programming** (if aiming for high-tier roles) or **Specialized Algorithm** study (e.g., String Algorithms, Advanced Graph Theory).	**Codeforces/TopCoder**, Books on **Advanced Algorithms** (Cormen/CLRS).
Launch Job Search	Leverage high DSA fluency to confidently approach any technical screen or interview round. Focus on **salary negotiation**.	**UpStride Job Search & Branding Architect Cohort** (Salary Negotiation), **Mock Interviews** (Specific to target companies/roles).