■ DSA Problem-Solving Roadmap

Phase 1 – Foundations (2-3 weeks)

- Brute Force Thinking
- Arrays Basics (Prefix Sum, Two Pointers, Sliding Window, Kadane’s Algorithm)
- Strings Basics (Hashmaps, Anagram check, Longest substring without repeating chars)
- Math & Bit Manipulation (GCD/LCM, Modular arithmetic, XOR tricks)

Phase 2 – Intermediate Problem Solving (3-4 weeks)

- Sorting + Searching (Binary Search variations, Rotated array search)
- Greedy Algorithms (Activity selection, Minimum coins, Huffman coding optional)
- Recursion & Backtracking (N-Queens, Subsets, Permutations, Sudoku Solver)
- Stacks & Queues (Next Greater Element, Min Stack, Sliding Window Maximum)

Phase 3 – Dynamic Programming & Advanced Arrays (4-6 weeks)

- Dynamic Programming (Fibonacci, Climbing Stairs, Knapsack, LCS, Edit Distance, Grid DP)
- Advanced Arrays & Heaps (Kth largest, Merge k sorted arrays, Heap problems)

Phase 4 – Graphs & Trees (4-5 weeks)

- Binary Trees & BSTs (Traversals, LCA, Diameter, Serialization/Deserialization)
- Graphs (BFS, DFS, Cycle detection, Shortest paths, Topological Sort, Union-Find)

Phase 5 – Advanced Topics (2-3 weeks, optional)

- Segment Trees / Fenwick Tree (Range queries + updates)
- String Algorithms (KMP, Rabin-Karp, Z-function, Trie)
- Special Techniques (Monotonic Stack, Bitmask DP, Advanced Graphs: MST, Floyd-Warshall)

■ Strategy to Follow

- Daily Practice: Solve 3–5 problems (mix easy, medium, hard).
- Use LeetCode (Easy/Medium → Hard) and Striver’s SDE Sheet / NeetCode 150.
- Focus on recognizing problem patterns (e.g., sliding window, two pointers).
- Revise solved problems regularly.