Data Structures & Algorithms Roadmap 2025

For Beginners Focused on Coding Interviews

1. Programming Foundations

• Learn a programming language (C++, Java, or Python).
• Understand variables, loops, functions, arrays, and recursion.
• Practice solving basic problems on platforms like LeetCode, Codeforces, or HackerRank.

2. Core Data Structures

Topic	Key Concepts
Arrays & Strings	Indexing, Sliding Window, Two Pointers
Linked Lists	Singly & Doubly Lists, Reversal, Fast-Slow Pointers
Stacks & Queues	Implementation, Expression Evaluation, Monotonic Stack
Hashing	Hash Maps, Collision Handling, Frequency Counting
Trees	Binary Trees, BST, Traversals, Depth/Height
Heaps	Min/Max Heap, Priority Queue, Heap Sort
Graphs	Adjacency List/Matrix, BFS, DFS, Shortest Path

3. Algorithms

• **Sorting & Searching:** Bubble, Merge, Quick, Binary Search
• **Recursion & Backtracking:** Subsets, Permutations, N-Queens
• **Dynamic Programming:** Fibonacci, Knapsack, LIS, DP on Strings
• **Greedy Algorithms:** Interval Scheduling, Huffman Encoding
• **Graph Algorithms:** Dijkstra, Floyd-Warshall, Union-Find
• **Divide & Conquer:** Merge Sort, Binary Search Variants

4. Problem Solving Strategy

• Start with easy problems, then move to medium difficulty.
• Learn patterns (e.g., Sliding Window, Binary Search on Answer).
• Track progress using spreadsheets or GitHub repos.
• Revisit and optimize older solutions.

5. Recommended Practice Platforms

• LeetCode (Interview preparation)
• GeeksforGeeks (Concepts + Practice)
• Codeforces (Competitive programming)
• HackerRank / HackerEarth (Beginner-friendly)
• NeetCode.io & Striver’s DSA Sheet (Structured learning path)

6. Final Tips

• Focus on understanding *why* a solution works, not just memorizing code.
• Discuss problems with peers and explain your thought process.
• Be consistent — aim for 1–2 problems per day.
• Build mini-projects that use data structures (e.g., LRU Cache, Graph Traversal Visualizer).