**Month 1: Basics of DSA & Core Data Structures**

The goal here is to understand fundamental data structures and basic problem-solving techniques.

**Week 1: Introduction to DSA & Complexity Analysis**

* **Topics:**
  + What is DSA?
  + Why learn DSA?
  + Time and Space Complexity (Big-O, Big-Theta, Big-Omega)
  + Recursion basics
* **Practice:**
  + Solve 10-15 easy problems on recursion and complexity.

**Week 2: Arrays & Strings**

* **Topics:**
  + 1D and 2D arrays (Basic operations)
  + Sliding window technique
  + Two-pointer approach
  + Introduction to Strings (basic manipulations)
* **Practice:**
  + Array problems (subarrays, maximum/minimum problems)
  + Solve 10 easy-medium string manipulation problems.

**Week 3: Linked Lists**

* **Topics:**
  + Singly and Doubly Linked Lists
  + Circular Linked List
  + Basic operations (insertion, deletion, reversal, etc.)
* **Practice:**
  + Implement linked lists from scratch.
  + Solve 5-7 problems on linked lists (reversal, palindrome check, cycle detection, etc.).

**Week 4: Stacks & Queues**

* **Topics:**
  + Stack and Queue data structures (implementation with arrays and linked lists)
  + Applications of stacks (infix to postfix, postfix evaluation, etc.)
  + Queue variants: Deque, Circular Queue, Priority Queue
* **Practice:**
  + Solve problems like balancing parentheses, stock span, largest rectangle in histogram.

**Month 2: Advanced Data Structures & Sorting/Searching Algorithms**

This month focuses on more complex data structures and algorithms that are key to solving harder problems.

**Week 5: Trees (Basic)**

* **Topics:**
  + Introduction to Trees, Binary Trees
  + Tree traversals (preorder, inorder, postorder, level-order)
  + Binary Search Trees (BSTs) and basic operations (insert, delete, search)
* **Practice:**
  + Implement tree traversals.
  + Solve 5-7 problems related to BSTs (find kth smallest element, validate BST, etc.).

**Week 6: Trees (Advanced)**

* **Topics:**
  + AVL Trees, Balanced Trees
  + Heaps (Min Heap, Max Heap) and Priority Queue
  + Trie data structure (basic)
* **Practice:**
  + Solve 7-10 problems on heaps (k largest elements, merge k sorted arrays, etc.).
  + Solve 2-3 easy problems with tries (autocomplete, prefix matching).

**Week 7: Sorting & Searching Algorithms**

* **Topics:**
  + Sorting algorithms (Selection, Bubble, Merge Sort, Quick Sort, Heap Sort)
  + Searching algorithms (Binary Search, Ternary Search)
  + Searching in 2D arrays
* **Practice:**
  + Implement and understand the time complexities of each sorting algorithm.
  + Solve 10 problems on binary search and 2D search.

**Week 8: Hashing**

* **Topics:**
  + Hash tables and Hash maps
  + Collision handling (chaining, open addressing)
  + Applications: Counting frequencies, finding duplicates, two-sum problem, etc.
* **Practice:**
  + Solve 5-7 hashing-based problems (two-sum, longest consecutive sequence, subarray sum, etc.).

**Month 3: Graphs, Dynamic Programming & Problem Solving**

In this month, focus on solving advanced problems that are frequently asked in interviews and coding contests.

**Week 9: Graphs (Basic)**

* **Topics:**
  + Representation of graphs (Adjacency matrix, adjacency list)
  + BFS (Breadth First Search)
  + DFS (Depth First Search)
  + Detecting cycles in graphs (directed and undirected)
* **Practice:**
  + Solve problems using BFS and DFS (connected components, island problems).

**Week 10: Graphs (Advanced)**

* **Topics:**
  + Dijkstra’s Algorithm
  + Bellman-Ford Algorithm
  + Topological Sorting
  + Minimum Spanning Trees (Kruskal, Prim’s algorithms)
* **Practice:**
  + Solve 5-7 problems related to shortest paths and MST.

**Week 11: Dynamic Programming (DP) Basics**

* **Topics:**
  + Introduction to DP, Memoization, Tabulation
  + Classic DP problems (Knapsack, Fibonacci, Coin Change)
  + Longest Common Subsequence (LCS)
* **Practice:**
  + Solve at least 10 beginner to intermediate DP problems (0/1 knapsack, LCS, etc.).

**Week 12: Advanced DP & Problem Solving**

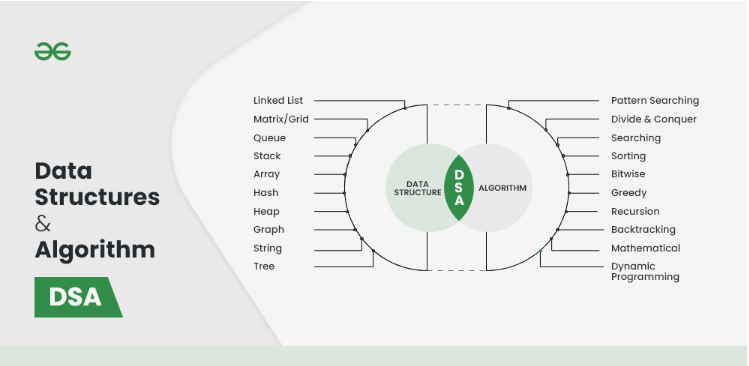
* **Topics:**
  + DP on trees
  + DP with bitmasks
  + Practice problem-solving with a mix of topics (graph + DP, etc.)
* **Practice:**
  + Solve 10-15 advanced DP problems.
  + Start solving problems from competitive programming platforms like Codeforces, LeetCode, etc.

**Key Resources:**

* **Platforms for Practice:** LeetCode, Codeforces, CodeChef, GeeksforGeeks, HackerRank
* **Books:**
  + "Cracking the Coding Interview" by Gayle Laakmann McDowell
  + "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein

**Final Notes:**

* **Daily Time Commitment:** ~2-3 hours/day



Data Structure:-

* + 1. Linked List
    2. Matrix/Grid
    3. Queue
    4. Stack
    5. Array
    6. Hash
    7. Heap
    8. Graph
    9. String
    10. Tree

Algorithm:-

1. Pattern Searching
2. Divide and Conquer
3. Searching
4. Sorting
5. Bitwise
6. Greedy
7. Recursion
8. Backtracking
9. Mathematical
10. Dynamic Programming