Phase 1: Foundations of Data Structures (4-6 Weeks)

Objective: Master basic data structures and their operations.

Week 1: Arrays and Strings

- **Topics:**
  - Array operations: Traversal, Insertion, Deletion, Merging.
  - String manipulations: Reversal, Anagram checks, Pattern matching.
- **LeetCode Problems (Easy to Medium):**
  - Two Sum, Maximum Subarray, Merge Sorted Array.
  - Longest Substring Without Repeating Characters, Group Anagrams.

Week 2: Linked Lists

- **Topics:**
  - Singly, Doubly, and Circular Linked Lists.
  - Operations: Insertion, Deletion, Reversal, Detecting Cycles.
- **LeetCode Problems:**
  - Reverse Linked List, Merge Two Sorted Lists, Detect Cycle in Linked List.

Week 3: Stacks and Queues

- **Topics:**
  - Stack: Push, Pop, Peek, Applications (Parentheses Matching, Min Stack).
  - Queue: Enqueue, Dequeue, Circular Queue.
  - Advanced Structures: Priority Queue, Deque.
- **LeetCode Problems:**
  - Valid Parentheses, Min Stack, Sliding Window Maximum.

Week 4: Hashing

- **Topics:**
  - HashMap, HashSet, Collision Handling.
  - Applications: Frequency counting, Subarrays with given sum.
- **LeetCode Problems:**
  - Two Sum, Subarray Sum Equals K, Longest Substring with At Most K Distinct Characters.

Week 5: Trees

- **Topics:**
  - Binary Trees: Traversals (Preorder, Inorder, Postorder, Level Order).
  - Binary Search Tree: Search, Insert, Delete, Balancing (AVL, Red-Black Trees).
  - Tree Problems: LCA, Diameter, Tree Serialization.
- **LeetCode Problems:**
  - Maximum Depth of Binary Tree, Validate Binary Search Tree, Diameter of Binary Tree.

Week 6: Heaps and Tries

- **Topics:**
  - Heaps: Min-Heap, Max-Heap, Priority Queue Applications.
  - Tries: Insert, Search, Auto-complete Applications.
- **LeetCode Problems:**
  - Kth Largest Element in an Array, Top K Frequent Elements, Implement Trie.

Phase 2: Core Algorithms (8-10 Weeks)

Objective: Understand and implement core algorithms.

Weeks 1-2: Sorting and Searching

- **Sorting Algorithms:** Bubble, Selection, Insertion, Merge, Quick, Heap.
- **Searching Algorithms:** Binary Search, Exponential Search.
- **LeetCode Problems:**
  - Find Peak Element, Search in Rotated Sorted Array, Merge Intervals.

Weeks 3-4: Recursion and Backtracking

- **Topics:**
  - Recursion Basics, Memoization.
  - Backtracking: N-Queens, Permutations, Subsets.
- **LeetCode Problems:**
  - Subsets, Combination Sum, Sudoku Solver.

Weeks 5-6: Graph Algorithms

- **Topics:**
  - Representations: Adjacency Matrix, Adjacency List.
  - Traversals: BFS, DFS.
  - Shortest Path: Dijkstra, Bellman-Ford, Floyd-Warshall.
  - Minimum Spanning Tree: Kruskal, Prim.
- **LeetCode Problems:**
  - Number of Connected Components, Shortest Path in a Grid.

Weeks 7-8: Dynamic Programming

- **Topics:**
  - DP Basics: Fibonacci, Climbing Stairs.
  - Patterns: Knapsack, Longest Common Subsequence, Palindromic Substrings.
- **LeetCode Problems:**
  - Longest Increasing Subsequence, Word Break, House Robber.

Phase 3: Advanced Topics and Practice (4-6 Weeks)

Objective: Solve diverse problems and simulate real-world scenarios.

Topics:

1. **Bit Manipulation:** Count set bits, XOR tricks, Subsets generation.
2. **Greedy Algorithms:** Activity Selection, Huffman Coding.
3. **Segment Trees and Fenwick Trees:** Range Queries.
4. **Advanced Graph Algorithms:** Topological Sorting, Maximum Flow.

LeetCode Problems:

- Hard problems from previous topics, e.g., Hard DP, Graphs, and Trees.

Daily Routine:

Time Allocation (2-4 hours daily):

1. **Concepts (30-60 mins):**
    a. Learn or revise a new topic.
2. **Problem Solving (90-120 mins):**
    a. Solve 2-3 problems from LeetCode.
3. **Revision (30 mins):**
    a. Review mistakes, revisit challenging problems.

Weekly Goals:

- Complete a data structure/algorithm and its associated problems.
- Revise and write notes for key concepts.

This roadmap is ambitious and covers everything comprehensively. Let me know if you'd like additional help, such as daily schedules, resource links, or specific topic explanations!