FULL DSA COURSE GUIDE - KABIROSKY

Overview:

This study guide is designed for a comprehensive approach to mastering Data Structures and Algorithms (DSA) over three months. It balances theory, coding practice, and problem-solving while integrating real-world applications to build a strong foundation in DSA.

Month 1: Foundation Building

Week 1: Time & Space Complexity, Big O Notation

- **Introduction:** Complexity analysis helps evaluate the performance of algorithms. Big O, Big Theta, and Big Omega notations describe the worst, average, and best-case scenarios of algorithms in terms of time and space.
- **Key Subtopics:**
  - Time Complexity (Best, Average, Worst case)
  - Space Complexity
  - Big O, Big Theta, Big Omega
  - Amortized Analysis
- **Recommended Resources:**
  - Big-O Cheat Sheet
  - Video: "Time Complexity Analysis" (by Abdul Bari on YouTube)
  - Platform: GeeksforGeeks - Time Complexity
- **Practice Problems:**
  - Easy: Time Complexity
  - Medium: Running Time of Algorithms
  - Hard: Maximum Subarray Sum
- **Time Allocation:**
  - Learning: 6 hours
  - Practice: 6 hours
  - Revision: 2 hours
- **Review:** Revisit key subtopics, and revise complexity analysis problems.

Week 2: Arrays and Strings

- **Introduction:** Arrays and strings are fundamental linear data structures. They are used for storing elements in a contiguous block of memory.
- **Key Subtopics:**
  - Array operations (insertion, deletion, searching)
  - Two-pointer technique
  - Sliding window approach

    - String manipulation
- **Recommended Resources:**
    - Video: "Arrays in Data Structures" (by Jenny's Lectures)
    - Platform: LeetCode Arrays
    - Platform: GeeksforGeeks - Strings
- **Practice Problems:**
    - Easy: Two Sum
    - Medium: Longest Substring Without Repeating Characters
    - Hard: Minimum Window Substring
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours
- **Review:** Reinforce string and array manipulation problems, two-pointer technique.

Week 3: Linked Lists

- **Introduction:** Linked lists store elements dynamically. They allow efficient insertions/deletions.
- **Key Subtopics:**
    - Singly Linked List
    - Doubly Linked List
    - Circular Linked List
    - Fast & Slow pointers, cycle detection
- **Recommended Resources:**
    - Video: "Linked Lists in Detail" (by FreeCodeCamp)
    - Platform: GeeksforGeeks - Linked Lists
    - Platform: LeetCode Linked List
- **Practice Problems:**
    - Easy: Reverse Linked List
    - Medium: Add Two Numbers
    - Hard: Merge K Sorted Lists
- **Time Allocation:**
    - Learning: 6 hours
    - Practice: 8 hours
    - Revision: 2 hours
- **Review:** Practice problems with pointer manipulation and linked list operations.

Week 4: Stacks and Queues

- **Introduction:** Stacks follow LIFO, while queues follow FIFO. They are widely used in problem-solving, including recursion and breadth-first search.
- **Key Subtopics:**
    - Stack operations (push, pop, peek)
    - Queue operations (enqueue, dequeue)

    - Implementation using arrays and linked lists
- **Recommended Resources:**
    - Video: "Stacks and Queues" (by MyCodeSchool)
    - Platform: GeeksforGeeks - Stacks
    - Platform: LeetCode Stacks
- **Practice Problems:**
    - Easy: Valid Parentheses
    - Medium: Largest Rectangle in Histogram
    - Hard: Sliding Window Maximum
- **Time Allocation:**
    - Learning: 6 hours
    - Practice: 8 hours
    - Revision: 2 hours
- **Review:** Practice stack and queue manipulation problems.

Month 2: Intermediate Data Structures

Week 5: Hashing

- **Introduction:** Hashing allows efficient searching and retrieval using hash tables.
- **Key Subtopics:**
    - Hash Tables, Maps, Sets
    - Collisions and Resolution Techniques
    - Applications of hashing
- **Recommended Resources:**
    - Video: "Hashing Techniques" (by Jenny's Lectures)
    - Platform: LeetCode Hash Table
- **Practice Problems:**
    - Easy: Two Sum
    - Medium: Subarray Sum Equals K
    - Hard: Longest Consecutive Sequence
- **Time Allocation:**
    - Learning: 6 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 6: Trees

- **Introduction:** Trees represent hierarchical data and are essential for efficient searching and sorting.
- **Key Subtopics:**
    - Binary Trees, Binary Search Trees
    - Tree traversal techniques (Preorder, Inorder, Postorder)

    - Balanced Trees (AVL, Red-Black Trees)
- **Recommended Resources:**
    - Video: "Trees and Binary Search Trees" (by CS50 Harvard)
    - Platform: GeeksforGeeks - Binary Trees
- **Practice Problems:**
    - Easy: Binary Tree Inorder Traversal
    - Medium: Construct Binary Tree from Preorder and Inorder Traversal
    - Hard: Serialize and Deserialize Binary Tree
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 7: Heaps and Priority Queues

- **Introduction:** Heaps are used to maintain the maximum or minimum element efficiently.
- **Key Subtopics:**
    - Min-Heap, Max-Heap
    - Heap Sort
    - Priority Queues
- **Recommended Resources:**
    - Video: "Heaps and Priority Queues" (by Abdul Bari)
    - Platform: LeetCode Heap
- **Practice Problems:**
    - Easy: Kth Largest Element in an Array
    - Medium: Top K Frequent Elements
    - Hard: Find Median from Data Stream
- **Time Allocation:**
    - Learning: 6 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 8: Graphs

- **Introduction:** Graphs are used to represent networks and are crucial in solving routing problems.
- **Key Subtopics:**
    - Graph Representation (Adjacency List, Matrix)
    - BFS, DFS
    - Cycle Detection
- **Recommended Resources:**
    - Video: "Graph Algorithms" (by Tushar Roy)
    - Platform: GeeksforGeeks - Graphs
- **Practice Problems:**
    - Easy: Graph Traversal

    - Medium: Course Schedule
    - Hard: Word Ladder
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours

Month 3: Advanced Algorithms & Optimization

Week 9: Greedy Algorithms

- **Introduction:** Greedy algorithms solve problems by making the best local decision at each step.
- **Key Subtopics:**
    - Activity Selection Problem
    - Fractional Knapsack
    - Huffman Coding
- **Recommended Resources:**
    - Video: "Greedy Algorithms" (by Abdul Bari)
    - Platform: GeeksforGeeks - Greedy Algorithms
- **Practice Problems:**
    - Easy: Greedy Algorithm for Activity Selection
    - Medium: Job Scheduling
    - Hard: Minimum Cost to Connect Sticks
- **Time Allocation:**
    - Learning: 6 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 10: Divide and Conquer

- **Introduction:** Divide and conquer solves problems by breaking them down into smaller subproblems.
- **Key Subtopics:**
    - Merge Sort
    - Quick Sort
    - Binary Search
- **Recommended Resources:**
    - Video: "Divide and Conquer Algorithms" (by CS50)
    - Platform: LeetCode Divide and Conquer
- **Practice Problems:**
    - Easy: Binary Search
    - Medium: Merge Sort

    - Hard: Kth Smallest Element in a Sorted Matrix
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 11: Dynamic Programming (DP)

- **Introduction:** Dynamic Programming solves problems by breaking them into overlapping subproblems and storing solutions.
- **Key Subtopics:**
    - Memoisation and Tabulation
    - Classical DP Problems (Knapsack, LIS)
- **Recommended Resources:**
    - Video: "Dynamic Programming Simplified" (by Tushar Roy)
    - Platform: LeetCode Dynamic Programming
- **Practice Problems:**
    - Easy: Climbing Stairs
    - Medium: Longest Increasing Subsequence
    - Hard: Edit Distance
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours

Week 12: Backtracking and Recursion

- **Introduction:** Backtracking solves constraint satisfaction problems by exploring all possible configurations.
- **Key Subtopics:**
    - N-Queens Problem
    - Sudoku Solver
    - Subset Generation
- **Recommended Resources:**
    - Video: "Backtracking Explained" (by CS50)
    - Platform: LeetCode Backtracking
- **Practice Problems:**
    - Easy: Subsets
    - Medium: Combination Sum
    - Hard: N-Queens
- **Time Allocation:**
    - Learning: 8 hours
    - Practice: 8 hours
    - Revision: 2 hours

Final Week: Mock Interview Prep and Revision

- **Revise:** Revisit all the key topics.
- **Mock Interview Questions:**
    - Practice questions from LeetCode's "Top Interview Questions."
    - Focus on time-bound problem-solving.
- **Project:** Implement a project using DSA concepts. For example, build a dynamic task scheduler using heaps and graphs or develop a pathfinding algorithm using BFS and DFS in a game or map-routing application.

Additional Tips:

- **Competitive Programming:** Spend 1-2 hours per week on platforms like Codeforces and LeetCode to sharpen problem-solving speed.
- **Advanced Learning:** Post-DSA, explore system design and contribute to open-source projects to apply your knowledge.

This guide will ensure a strong command over DSA and problem-solving skills!