Here's a 6-month plan to master Data Structures and Algorithms (DSA) systematically, covering all key concepts:

Month 1: Basics of Programming and Mathematics

Goals:

- Brush up on basic programming skills.
- Build a strong mathematical foundation.

Topics:

1. **Programming Language Basics:** Choose a language like Python, C++, or Java.
   - Input/Output
   - Loops, Conditional Statements
   - Functions and Recursion
2. **Mathematics for DSA:**
   - GCD, LCM, Modular Arithmetic
   - Prime Numbers, Sieve of Eratosthenes
   - Basic Combinatorics (nCr, permutations)

Practice:

- Platforms: HackerRank, Codewars
- Solve 20–30 basic problems on loops, recursion, and prime numbers.

Focus: Programming foundations, simple logic-based problems, and mathematical concepts.

Questions:

1. **Basic Programming:**
   - Write a program to check if a number is prime.
   - Sum of digits of a number.
   - Reverse a string.
   - Factorial using recursion.
2. **Mathematics:**
   - GCD of two numbers.
   - LCM of two numbers.
   - Sieve of Eratosthenes for primes.
   - Number of trailing zeros in factorial.

Month 2: Arrays, Strings, and Linked Lists

Goals:

- Master fundamental data structures.
- Solve problems on linear data structures.

Topics:

1. **Arrays:**
   - Basics, Prefix Sums, Sliding Window
   - Two Pointers, Kadane’s Algorithm
2. **Strings:**
   - Palindromes, Anagram Detection
   - String Manipulations (e.g., reversing words)
3. **Linked Lists:**
   - Single/Doubly Linked Lists
   - Operations: Insertion, Deletion, Reversal

Practice:

- Platforms: LeetCode, GeeksforGeeks
- Solve 50 problems: 20 on Arrays, 15 on Strings, 15 on Linked Lists.

Focus: Linear data structures and their operations.

Questions:

1. **Arrays:**
   - Find the maximum sum of a subarray (Kadane's Algorithm).
   - Rotate an array.
   - Find duplicates in an array.
   - Trapping Rain Water.
2. **Strings:**
   - Check if a string is a palindrome.
   - Find the first non-repeating character.
   - Group anagrams.
   - Longest Palindromic Substring.
3. **Linked Lists:**
   - Reverse a linked list.
   - Merge two sorted linked lists.
   - Detect a cycle in a linked list.
   - Remove nth node from the end.

Month 3: Stacks, Queues, and Hashing

Goals:

- Understand non-linear linear data structures.
- Solve medium-level problems.

Topics:

1. **Stacks and Queues:**
   - Applications: Balanced Parentheses, Next Greater Element
   - Implementations: Stack using Array/Queue
2. **Hashing:**
   - Hash Maps and Hash Sets
   - Problems on frequency counting, subarray sums

Practice:

- Platforms: CodeChef, NeetCode
- Solve 40 problems: 15 on Stacks/Queues, 25 on Hashing.

Focus: Non-linear data structures and hashing.

Questions:

1. **Stacks and Queues:**
    - Evaluate Reverse Polish Notation.
    - Next Greater Element.
    - Valid Parentheses.
    - Implement a queue using stacks.
2. **Hashing:**
    - Two Sum.
    - Subarray Sum Equals K.
    - Longest Consecutive Sequence.
    - Top K Frequent Elements.

Month 4: Trees and Graphs

Goals:

- Understand hierarchical data structures.
- Solve traversal and connectivity problems.

Topics:

1. **Trees:**
    - Binary Trees, Binary Search Trees (BST)
    - Tree Traversals: BFS, DFS
    - Problems: Diameter of Tree, Lowest Common Ancestor
2. **Graphs:**
    - Representations: Adjacency List/Matrix
    - BFS, DFS, Shortest Path (Dijkstra's, Bellman-Ford)
    - Minimum Spanning Tree (Prim's, Kruskal's)

Practice:

- Platforms: LeetCode, Codeforces
- Solve 30 problems: 20 on Trees, 10 on Graphs.

Focus: Hierarchical data structures and graph traversal techniques.

Questions:

1. **Trees:**
    - Binary Tree Inorder Traversal.
    - Maximum Depth of Binary Tree.
    - Diameter of Binary Tree.
    - Lowest Common Ancestor of a Binary Tree.
2. **Graphs:**
    - Clone a graph.
    - Number of Islands.
    - Shortest Path in Binary Matrix.
    - Detects cycle in a directed graph.

Month 5: Advanced Concepts

Goals:

- Learn advanced data structures and algorithms.

Topics:

1. **Heaps and Priority Queues:**
   - Heap Sort, Applications in Graphs
2. **Divide and Conquer:**
   - Merge Sort, Quick Sort
3. **Dynamic Programming (DP):**
   - Basics of Memoization and Tabulation
   - Problems: Knapsack, Longest Increasing Subsequence

Practice:

- Platforms: AtCoder, Codeforces
- Solve 50 problems: 30 on DP, 10 on Heaps, 10 on Divide and Conquer.

Focus: Heaps, Divide and Conquer, and Dynamic Programming.

Questions:

1. **Heaps:**
   - Kth Largest Element in an Array.
   - Find Median from Data Stream.
   - Merge K Sorted Lists.
2. **Divide and Conquer:**
   - Merge Sort implementation.
   - Find the Peak Element.
3. **Dynamic Programming:**
   - Longest Increasing Subsequence.
   - 0/1 Knapsack Problem.
   - House Robber.
   - Longest Common Subsequence.

Month 6: Problem-Solving and Revision

Goals:

- Consolidate your learning.
- Focus on competitive programming problems.

Tasks:

1. **Solve Mixed Problems:**
   - Mix topics like Trees + DP, Graphs + Greedy.
2. **Mock Contests:**
   - Participate in coding competitions (e.g., on Codeforces, LeetCode Weekly).
3. **Revise Weak Areas:**
   - Identify and revise your weaker topics.

Practice:

- Solve 60 problems: 10 on each core topic (Arrays, Strings, Trees, Graphs, DP, etc.).

Focus: Comprehensive problem-solving across all topics.

Questions:

1. **Mixed Problems:**
   - Course Schedule (Graph + DFS).
   - Word Break (DP).
   - Largest Rectangle in Histogram (Stack).
   - Median of Two Sorted Arrays (Divide and Conquer).
   - Trapping Rain Water (Array + Two Pointers).
2. **Mock Contests:**
   - Participate in weekly contests on LeetCode or Codeforces.

Resources:

- **Courses:**
  - CS50 by Harvard (Free)
  - Coding Ninjas DSA Course
  - NeetCode DSA Roadmap
- **Books:**
  - *"Introduction to Algorithms"* by Cormen (CLRS)
  - *"Cracking the Coding Interview"* by Gayle Laakmann McDowell
- **YouTube Channels:**
  - CodeWithHarry
  - Abdul Bari
  - Love Babbar DSA Sheet

Tips:

1. **Daily Practice:** Spend at least 2 hours solving problems.
2. **Consistency:** Stay disciplined and follow the schedule.
3. **Debugging:** Understand your mistakes thoroughly and learn from them.

Follow **@pythonboyz** for more ❤️