DATA STRUCTURES AND ALGORITHMS SUGGESTIONS BY @tech.boy.deepak

1. Introduction to Programming

Language Basics: Variables, data types, operators, conditional statements, loops

Simple Programs: Writing small programs to understand the syntax and basic programming concepts

2. Basic Data Structures

Arrays: Introduction, traversal, insertion, deletion, searching (linear and binary search), sorting (bubble, selection, insertion sort)

Linked Lists: Singly linked lists, doubly linked lists, circular linked lists, operations (insertion, deletion, traversal)

Stacks: Introduction, operations (push, pop, peek), implementation using arrays and linked lists

Queues: Introduction, operations (enqueue, dequeue, front, rear), types (linear, circular), implementation using arrays and linked lists

3. Recursion

Understanding Recursion: Concept, base case, recursive case

Examples: Factorial, Fibonacci series, Tower of Hanoi, binary search using recursion

4. Intermediate Data Structures

Trees: Introduction, binary trees, traversal (in-order, pre-order, post-order), binary search trees (BST), operations in BST

Heaps: Introduction, types (max heap, min heap), operations (insertion, deletion, heapify), heap sort

Hashing: Introduction, hash function, collision handling methods (chaining, open addressing)

5. Advanced Data Structures

Graphs: Introduction, representation (adjacency matrix, adjacency list), traversal (BFS, DFS)

Tries: Introduction, implementation, applications

Segment Trees: Introduction, construction, update, range queries

Fenwick Trees (Binary Indexed Trees): Introduction, update, prefix sum queries

6. Algorithms

Sorting Algorithms: Quick sort, merge sort, heap sort, counting sort, radix sort

Searching Algorithms: Depth-first search (DFS), breadth-first search (BFS), Dijkstra's algorithm, A* algorithm

Dynamic Programming: Introduction, memoization, tabulation, examples (knapsack problem, longest common subsequence, longest increasing subsequence)

Greedy Algorithms: Introduction, examples (activity selection, Huffman coding, Kruskal's algorithm)

Backtracking: Introduction, examples (N-Queens problem, Sudoku solver, permutation generation)

Divide and Conquer: Introduction, examples (merge sort, quick sort, Karatsuba algorithm for multiplication)

7. Advanced Topics

String Algorithms: String matching algorithms (KMP, Rabin-Karp), tries, suffix arrays

Graph Algorithms: Minimum spanning tree (Prim's, Kruskal's algorithm), shortest path algorithms (Dijkstra's, Bellman-Ford, Floyd-Warshall), topological sort, strongly connected components

Computational Geometry: Convex hull, line intersection, closest pair of points

Network Flow: Maximum flow (Ford-Fulkerson algorithm), minimum cut

8. Practice and Application

Practice Problems: Solving problems on platforms like LeetCode, HackerRank, CodeChef, and GeeksforGeeks

Projects:

Implementing data structures and algorithms in real-world projects or simulations

9. Advanced Study

Competitive Programming: Participating in contests to improve problem-solving speed.

My suggestions :

If you are a beginner you have to learn c programming for 2 months

from : https://youtu.be/irqbmMNs2Bo?si=eIdXaBTm9_Rd09GO

from : https://youtu.be/HdvRHC5TiwE?si=FGYOvJ_6exu4eFUo

If you are good with c :

And for c++ and DSA : https://youtu.be/EAR7De6Goz4?si=yrsQZKKNC5t-ICP4

And for Java : https://youtu.be/rZ41y93P2Qo?si=dSMUfX2wpWX2o0o6

And for Python : https://youtu.be/pkYVOmU3MgA?si=Rjtu1on_1RwRjFqm

For practicing Problems :

Use Geeks for Geeks Website OR Hacker Rank

Use Leetcode for Advance Problems