Session 1: Analysis of Algorithms, Mathematics & Bit Magic

Analysis of Algorithms

- Understanding Order of Growth and how input size affects performance
- Best, Average, and Worst-case behavior of algorithms
- Working with Asymptotic Notations: Big-O, Big-Theta, Big-Omega
- Applying time analysis to common algorithmic patterns

Mathematics

- Strengthening mathematical foundations with coding-friendly problems
- Solving basics like Factorial computation, HCF, and LCM
- Understanding the Sieve of Eratosthenes and prime-generation logic

Bit Magic

- Learning how Bitwise Operators work (AND, OR, XOR, NOT, Shifts)
- Understanding bit tricks through important example problems
- Applying bitwise logic to optimize computations

Session 2: Recursion, Arrays & Searching

Recursion

- Understanding recursive thinking and base cases
- Tail recursion and its optimization benefits
- Practicing problems such as Rope Cutting, Tower of Hanoi, Josephus Problem

Arrays

- Introduction to arrays, indexing, and operations
- Working through real problems like Stock Buy & Sell, Trapping Rain Water
- Memory layout and common pitfalls

Searching

- Understanding Binary Search technique and how it improves performance
- Analyzing binary search time complexity
- Solving various problems using binary search patterns

Session 3: Sorting, Matrix & Hashing

Sorting

- Learning comparison-based sorting techniques
- Understanding Insertion Sort, Quick Sort, Radix Sort
- Knowing when to use which sorting method

Matrix

- Working with matrix traversal and manipulation
- Solving example-based problems like Snake Pattern and Spiral Traversal
- Understanding 2D array memory mapping

Hashing

- Introduction to hashing and time complexity analysis
- Use cases of hashing in fast lookup/search
- Learning Direct Address Table and its applications

Session 4: Strings & Linked Lists

String

- Introduction to strings and basic operations
- Solving advanced pattern-matching algorithms:
    - Rabin-Karp Algorithm
    - KMP Algorithm
- Understanding efficient substring search

Linked List

- Learning Singly, Doubly, and Circular Linked Lists
- Solving core problems like loop detection, LL intersection
- Implementing LRU Cache using linked list logic

Session 5: Stack, Queue, Deque & Trees

Stack

- Introduction, implementation, and stack properties
- Solving Balanced Parentheses, Stock Span, Infix/Prefix/Postfix problems

Queue

- Understanding Queue operations and real-world usage
- Problems like Reversing a Queue, Generate Numbers with Given Digits

Deque

- Understanding Deque structure and implementation

- Solving Sliding Window Maximum, First Circular Tour problems

Tree

- Working with tree basics and traversal methods
- Solving problems from Binary Tree Height, Level-order Traversal
- Exploring advanced problems: Burn a Tree, Serialize/Deserialize Binary Tree

Session 6: BST, Heap & Graphs

Binary Search Tree

- Understanding BST operations: search, insert, delete
- Solving BST problems like Check for BST, Vertical Sum

Heap

- Learning Min-Heap, Max-Heap, and Priority Queue
- Working on heap-based problem scenarios

Graph

- Understanding Graph Representation: adjacency list/matrix
- Implementing BFS and DFS
- Learning key algorithms: Prim’s, Dijkstra’s, Kosaraju’s
- Solving graph-based tutorials and problems

Session 7: Greedy, Backtracking & Dynamic Programming

Greedy Algorithms

- Learning the greedy approach to optimization problems

- Solving tasks like Activity Selection, Fractional Knapsack

Backtracking

- Understanding backtracking principles and recursion depth
- Solving problems like Rat in a Maze, N-Queen, Sudoku Solver

Dynamic Programming

- Learning DP with memoization and tabulation
- Solving LCS, Coin Change, LIS, Egg Dropping problems

Session 8: Trie, Segment Tree & Disjoint Set

Trie

- Understanding Trie representation, insert, search, delete
- Solving Trie-related problems

Segment Tree & Binary Indexed Tree

- Learning range-query structures: Segment Tree, Fenwick Tree
- Solving range min/max/sum problems

Disjoint Set

- Understanding DSU operations: Find, Union
- Learning Union by Rank and Path Compression
- Applying DSU to classical graph and connectivity problems

Session 9: Projects (Implementation & Visualization)

Hands-on Projects

- **Sudoku Solver** – backtracking-based solution
- **Shortest Path Finder** – graph algorithms in action
- **Tic Tac Toe** – logic-based implementation
- **N-Queen Visualizer** – illustrating recursive solutions
- **Binary Tree Visualizer** – tree construction and traversal animations