Competitive Programming

Pre-requisites

- Basic Knowledge of Programming (any language)

Course Highlights

- Dedicated live doubt solving sessions by mentor.
- 24x7 Support on Teams for your doubt solving.
- Topic Wise Real World Competitive Programming Practice Questions
- Easy to Hard Problem sets from leet code, hacker earth, hacker rank.
- Cover all corner test cases/invalid inputs for possible undetectable errors.
- Cracking Coding Interviews

Part-1: Python Programming

Part-2: Data Structure & Algorithms

Part-3: Competitive Problem Solving

Python Programming

- Writing and Executing Scripts in Python
- Input Output Operations in Python
- Data Type and Data Structures in Python
    - Number, String, List, Tuple, Dictionary, Set, Frozen Set
- Operators in Python
    - Arithmetic, Comparison, Logical, Bitwise, Membership, Identity
- Control Statements in Python
    - If-else, nested if else, multiple condition in python
- Loops in Python
    - For, While Loop, continue, break, else statement with loop

- Functions in Python
  - Scope, Recursion, Decorators, Generators, lazy Evaluation, map, lambda, reduce, filter
- OOPs using Python
  - Class, Object, Inheritance, Duck Typing, Abstraction, Encapsulation, Abstract Class
- Exception Handling in Python
- Standard Library of Python

Data Structures and Algorithms

- Introduction Data Structure and Algorithms
  - Operations, Concepts, Design Principle
- Algorithms: Complexity, Time Space Tradeoff
- Mathematical Notation and Functions
- Complexity of Algorithms
- String Processing
- Arrays
  - One, Two, Three dimensional Arrays
  - Polynomial Representation, addition
  - Sparse Matrix & Matrix Operations
  - Array Operations
- Linked Lists
  - Singly, Doubly and Circular Link List
  - Polynomial Representation
  - Operations of Chains
- Stack, Ques, Recursion
  - Stack using Array & Link List
  - Arithmetic Expressions; Polish Notation
  - Applications of Stacks
  - Recursion, Towers of Hanoi
  - Recursive Procedures by Stacks
  - Ques, Evaluation of Expressions
  - Linked Representation of Ques
  - Deques & Priority Queues
- Hashing
  - Hash Tables, Hash functions
  - Dynamic Hashing
  - Dynamic Hashing using Directories
  - Dynamic Hashing without Directories
- Trees
  - Binary Tree Representation in Python
  - Binary Tree Traversals – Inorder, Preorder, Postorder
  - Searching and Inserting in Binary Search Tree
  - BFS and DFS

    - Deleting in a Binary Search Tree
    - AVL Search Tree, B-tree
    - Huffman's Algorithm
    - Red-Black Tree, Rotations, Insertion, Deletion
- Heaps
    - Binomial Heap, Operations
    - Fibonacci Heap
    - Symmetric Min-Max Heaps
- Graphs Theory
    - Sequential Representation of Graphs
    - Adjacency Matrix, Path Matrix
    - Wars hall's Algorithm, Shortest Path
    - Traversal of a Graph
    - Depth First Search, Breadth First Search, Spanning Trees
    - Minimum Cost Spanning Trees – Kruskal's, Prim's, Sollin's Algorithm
    - Dijkstra's algorithm
    - Activity Networks, AOV (activity on vertex) and AOE (activity on edge)
- Sorting & Searching Algorithms
    - Insertion Sort
    - Quick Sort
    - Merge Sort
    - Heap Sort
    - Radix Sort
    - Bucket Sort
- Dynamic Programming
    - Assembly-line scheduling
    - Matrix-chain multiplication
    - Elements of dynamic programming
    - Longest Common Subsequence (LCS)
    - Optimal Binary Search Trees
- Greedy Algorithms
    - An activity-selection problem
    - Elements of the greedy strategy
    - Huffman codes
    - A task scheduling Problem

Competitive Problem Solving

- Python Programming Competitive Problem Solving
- Competitive Questions on Graph Theory, Disjoint Set Union, Minimum Spanning Tree
- Segment Tree, String Algorithms, Trees Based Competitive Questions
- Competitive Questions Based on Graph Coloring, Network Flow