Data Structures and Algorithm Design

Online Course

Course Focus: Preparing students to crack coding interviews of all product companies which focus on Coding DSA skills like Google, Microsoft, Amazon, Adobe, Facebook, LinkedIn etc.

- **Skills : Master Algorithm Design, Data Structures and Problem Solving Skill**
- Become an expert problem solver with 80+ hours targeted course.
- Get ready to crack product companies like Google, Amazon, Microsoft and high paying start-ups.
- Learn Advance Coding concepts and get ready for the competitions like ACM-ICPC, Google Codejam and more..
- Dedicated Live Doubt Clearing Session from Course Mentors only.
- Take an edge over other students and grab highest job offer in-campus or off-campus.
- Interview preparation to crack coding interviews.
- **Language of Course :** C/C++/JAVA/Python : Complete Focus on DSA and problem solving, So programming language does not make any difference. Student can code in any language.
- **Language of Communication :** ENGLISH + HINDI
- 120+ Handpicked Top Problems : 80+ Hours Video Content
- 250+ Problems Assignment : Total 400+ problems in course video and assignment that make ure 95% problems asked in Coding interviews are already practiced by you.
- **LEARN AT YOUR PACE :** You have access to course videos for 6 Months + 1 Months that gives you flexibility to learn at your own pace and do a lot of practice to become master in DSA.

Course Curriculum:

Data Structures

- Arrays
- Strings
- Matrices
- Stacks
- Queues/ Monotonic Queues
- Linked List
- Binary Tree/ BST
- Segment Tree
- Trie
- Priority Queues/Heaps
- Hashes
- Graphs

Algorithms

- Understanding complexity(time/space) of an algorithm/course

- Big O notation
- Brute Force Approach
- Backtracking/ Recursion
- Sliding Window
- Trial & Error Methods
- **Binary search algorithm and when to apply**
    - Learning concept and learn to identify that a problem can be solved using BS
    - Tips to avoid edge cases and overflow and underflow
- **Merge sort & Quick sort and what problems can be solved with them?**
    - Concept and how things change at scale
    - How to optimize further with tricks?
- **Dynamic programming approach : How to think in DP?**
    - How to identify a problem to be a DP problem
    - How to change recursive to DP
- **Greedy approach**
    - Which problems can be solved using greedy method
    - How to get complexity of it
- **Backtracking approach**
    - What template all problems follow?
    - How to fit your problems in that template?

Optimization of Code

- Understanding complexity(time/space) of an algorithm/program
- Big O notation
- Brute Force Approach
- Backtracking/ Recursion
- Learning concept and learn to identify that a problem can be solved using BS
- Tips to avoid edge cases and overflow and underflow
- Concept and how things change at scale
- How to optimize further with tricks?
- How to identify a problem to be a DP problem
- How to change recursive to DP
- Which problems can be solved using greedy method
- How to get complexity of it
- What template all problems follow?
- How to fit your problems in that template?

Best Coding Practices

- Understanding execution of recursive functions
- How to scale a solution once you got it right?
- Improving time & space complexity
- Corner cases
- Invalid inputs
- Recursion termination conditions
- Logical errors
- Memory Leaks
- Test cases

Cracking Coding Interviews

- How to start solutioning?
- Do and Don’ts in interviews

- What are bare minimum for a solution?
- When to start optimizations?
- When to think of scale and complexity analysis?