Preparing for Software Engineer roles @ Product-based companies in 100 days

InterviewPrep.AppliedCourse.com

Resources

- Books: CLRS (Introduction to Algorithms)
- Practice Problems
  - Leetcode (Best for Interview preparation)
  - Hacker rank
  - CodeForces
  - CodeChef
  - And many others…….
- Try to solve ~ 500 problems for an average student

Phases of Learning

- Programming [1-3 days to recap, assuming you know programming]
- DS & Algo: Concepts + Easy problems [4-60]
- Problem Solving: Try different “patterns” of problems (medium to hard) [60-90]
- Advanced DS [90-95]
- Misc topics [95-100]

Write executable code

- Read the problem statement
- Pseudo-code/Logic
- Time and Space Complexity
- Code it up in Python/Java/C++/C/any-major-language
- Handle all boundary cases

DAY 1 - DAY 3

- Revise the programming concepts (c/ c++/Java/python)
- Important things to revise
  - C/C++ : Pointers
  - C++: STL [good for finance companies like DE Shaw]
  - Python: Data structures (list, tuple, dict, set, etc) and Basics of OOP [Very popular]
  - Java: Libraries and Basic OOP concepts
- Python Practice Questions: https://edabit.com/challenges
- For Non-CS (start with easy) and CS (start with medium)
- Practice at least 50 problems

DAY 4

- Algorithmic Complexity: Big O, Theta, Omega
- Analyse time and space complexity
  - For loops, Nested For loops
  - For loops with breaks
  - Recursion: Tree based methods, Master Theorem
  - *Space Complexity: Ignore input and output space.
- Revise these concepts. We will encounter examples all throughout problem solving.

DAY 5- DAY 25

- Basics Data structures [4 days per topic]
  - Arrays
  - Linked Lists
  - Stack
  - Queue
  - Strings
- Time & Space Complexity for key operations
- When to use what?
- Practice easy Problems of each topic
  - 15 Problems (links in the next slide)

DAY 5- DAY 25

- Practice Easy problems
  - Arrays: https://leetcode.com/tag/array/
  - Linked lists: https://leetcode.com/tag/linked-list/
  - Stacks: https://leetcode.com/tag/stack/
  - Queues: https://leetcode.com/tag/queue/
  - Strings: https://leetcode.com/tag/string/

DAY 26 - DAY 40

- Algorithms
  - Searching and Sorting Algorithms (3 Days)
  - Divide and Conquer algorithms (2 Days)
  - Greedy Algorithms (3 Days)
  - Dynamic Programing (7 Days)
- Practice Easy problems to each topic
  - 15-20 problems for each topic (links in the next slide)

DAY 26 - DAY 40

- Practice Easy Problems
  - Searching: https://leetcode.com/tag/binary-search/
  - Sorting: https://leetcode.com/tag/sort/
  - Greedy: https://leetcode.com/tag/greedy/
  - Dynamic Programming: https://leetcode.com/tag/dynamic-programming/

DAY 41 - DAY 60

- Non-Linear Data structures
  - Trees (8 Days)
    - Binary Tree
    - Binary Search Tree
    - AVL
- Heaps (2 Days)
- Hashing (2 Days)
- Graphs (5 Days)
- Back Tracking (3 Days)

DAY 41 - DAY 60

- Practice Problems links
  - Trees: https://leetcode.com/tag/tree/
  - Heaps: https://leetcode.com/tag/heap/
  - Hashing: https://leetcode.com/tag/hash-table/
  - Graphs: https://leetcode.com/tag/depth-first-search/
    - https://leetcode.com/tag/breadth-first-search/
  - Backtracking: https://leetcode.com/tag/backtracking/

DAY 61 - DAY 90

- Focus on Problem Solving
- Some of the patterns for coding problems as follows
  - Fast and Slow Pointers
    - Examples: [Google problem name + leetcode]
      - Linked List Cycle
      - Middle of linked list
      - Happy Number

DAY 61 - DAY 90

- Two pointers
  - Examples:
    - Find pair with target sum
    - Squaring a sorted array
    - Find Triplet sum equals to zero
    - Dutch National Flag Algorithm

DAY 61 - DAY 90

- In place reversal of linked list
  - Examples:
    - Reverse Linked List
    - Reverse a Sub list
    - Reverse every n-element sub list

DAY 61 - DAY 90

- Breadth First Search
  - Examples:
    - Level Order traversals
    - Zigzag Traversal
    - Connect level order siblings
    - Level order successor

DAY 61 - DAY 90

- Depth First Search
  - Examples:
    - Maximum Depth of Binary Tree
    - Number of Islands
    - Critical connections in a network
    - Clone Graph
    - Path Sum

DAY 61 - DAY 90

- Bitwise XOR
  - Examples:
    - Single Number
    - Two single Numbers
    - Counting bits

DAY 61 - DAY 90

- Two Heaps
  - Examples:
    - Find the median of a number stream
    - Sliding window median
    - Maximize capital

DAY 61 - DAY 90

- Modified Binary Search
  - Examples:
    - Median of Two Sorted Arrays
    - Ceiling of a number
    - Search in a sorted infinite array
    - Bitonic array maximum

DAY 61 - DAY 90

- Top k - elements
  - Examples:
    - Kth smallest element
    - Connect ropes
    - Kth Largest Element in a Stream
    - K-closest numbers

DAY 61 - DAY 90

- K - Way merge
  - Examples:
    - Merge K Sorted Lists
    - Kth Smallest Number in M Sorted Lists
    - Kth Smallest Number in a Sorted Matrix
    - Smallest Number Range

DAY 61 - DAY 90

- 0/1 Knapsack (Dynamic programming)
  - Examples:
    - Equal subset sum partition
    - Minimum subset sum Difference
    - 0/1 knapsack

DAY 61 - DAY 90

- Topological sort
  - Examples:
    - Tasks Scheduling
    - Tasks Scheduling Order
    - Alien Dictionary

DAY 61 - DAY 90

- Subsets
  - Examples:
    - Balanced parenthesis
    - Subsets with duplicates
    - Permutations

DAY 61 - DAY 90

- Merge Intervals
  - Examples:
    - Merge Intervals problem
    - Insert Interval
    - Intervals Intersection
    - Conflicting Appointments

DAY 61 - DAY 90

- Sliding Window
  - Examples:
    - Longest Substring Without Repeating Characters
    - Sliding Window Maximum
    - Minimum Window Substring
    - Number of Submatrices That Sum to Target

DAY 61 - DAY 90

- MinMax
  - Examples:
    - Guess Number Higher or Lower
    - Stone Game
    - Guess the word

DAY 91 - DAY 95

- Advanced Data structures
  - Tries
  - Red black Trees
  - B-Tree and B+ Trees
  - Disjoint sets
  - Segment Trees

DAY 96 - DAY 100

- Understand Computational complexity theory: NP-completeness & NP hardness.
- Knapsack problem.
- Travelling salesman problem.

Write executable code

- Read the problem statement [3-5 mins]
- Pseudo-code/Logic [5-7 mins]
- Time and Space Complexity [2 mins]
- Code it up in Python/Java/C++/C/any-major-language [10-15 mins]
- Handle all boundary cases [while coding]

Ideal: 20-25 min per problem, especially easy and medium problems.

InterviewPrep.AppliedCourse.com

- Concepts (DS & Algo) Explanation: ~90 hrs
- Solved problems + Video explanations: 210 [adding more]
- Covers all the major “patterns”
- Practice problems after each solved problem: 2-3
- Query resolution: 5-6 hrs (Max: 24 hrs)
- Monthly practice/assessment exams
- Mock interviews after assessment tests.
- Placement Prep and Job assistance.