@learn.machinelearning

GET A JOB AT

Part - 2

HOW TO PREPARE

@learn.machinelearning

- It’s important for any engineer, even the senior ones, to brush up on their coding skills and algorithms. Bear in mind that data structures and algorithms are the bread and butter of software engineering. You should know the data structure inside out, and what algorithms tend to go along with each data structure.
- Trees, trees and trees!
  - You should definitely brush up on the basic tree construction, conversion, traversal and ma nipulation algorithms.
  - You should be very much comfortable in binary trees, BSTs, and Trie related questions.
  - Familiarise yourself with the concept of n-ary trees and know at least one balanced version: AVL or red black tree.
  - Review recursive as well as iterative approaches.

HOW TO PREPARE

- Graphs
  - Graphs are the most fundamental and flexible way of representing any kind of a relationship.
  - Many questions can be visualised as a graph problem, so this should be the first data structure to come to your mind.
  - You should know the basic graph traversal algorithms: breadth-first search (BFS) and depth-first search (DFS). You should know their computational complexity, their tradeoffs (BFS vs DFS), and how to implement them in real code.
  - Try to study up on algorithms, such as Dijkstra and Prim.
  - Topological sorting and cycle detection are yet another important topics.

HOW TO PREPARE

- Hash Tables
- Arrays and Strings
  - Palindromes
  - Anagrams
  - Parentheses
  - Subsequence
  - Subarray
  - etc..
- Recursion and Backtracking
- Memoization and Greediness
  - LCS, LIS, Min cost path, 0–1 Knapsack and Palindrome partitioning
  - Greedy algorithms: Some problems around Greediness that are mostly asked are Activity Selection, Maximise difference in circular array, K bookings and Fractional knapsack.

HOW TO PREPARE

- Big O Notation
- Searching and Sorting
- Linked Lists
    - loop detection, cloning, flattening, union, intersection and basic manipulation.
    - Also read about circular and doubly linked lists.
- Stacks and Queues
    - Queue using Stacks, Stack using Queues, K stacks in an array.
    - Stock span, celebrity problem, balanced parentheses, NGE, Histogram problem
- Heaps
- Others...........

IMPORTANT TIPS

- Picking a programming language
  - Before anything else, you need to pick a programming language for your algorithmic coding interview. You’re gonna spend majority of your time writing code in an interview so make sure you are pretty familiar with at least one programming language.
- Practice coding by hand
- Practice writing code in the most realistic way possible
- Increase difficulty gradually
- Study your chosen programming language
- Data structures are your weapons
- Know your resume
- Mock Interviews

WHAT INTERVIEWERS ARE LOOKING FOR IN THE CODING CHALLENGE?

- Coding skills
- Problem-solving and creative-thinking ability in a short time frame
- Ability to solve problems in a structured and systematic way
- Communication skills — clarity of thought
- How you handle feedback
- Knowledge of optimisation tradeoffs between space and time complexity
- Testing code as you write it

HOW TO SOLVE A QUESTION

@learn.machinelearning

- Read and analyse the problem
- Ask clarifying questions — This is absolutely expected!
- Explain your approach — Do not start coding right away!
  - Articulate your thoughts,Communicate your assumptions,Ask for help,Analyse the time and space complexity of your potential solution,Keep thinking,Manage Your Time Effectively,Be honest and admit you don’t know
- Once you get a Green light — Write readable and well-factored code
  - Use your strongest coding language,Write helper functions, Code the edge cases,Write your code in step fashion
- Take a walk — Now is your chance to find bugs!
- Test the code

TYPICAL BEHAVIORAL QUESTIONS

- What could you have done better?
- What kind of technologies are you most excited about?
- What were some excellent collaborations you've had?
- This involves you explaining to them how you have collaborated with others in your past projects.
- What did you learn from your internship experience? What challenges did you face?
- How can we help you achieve your career ambitions?
- Describe a situation in which you failed. What did you learn?
- Give me an example of a difficult problem you solved. How did you solve this problem?
- Can you tell me about a time you set and achieved a certain goal?Give an example of a goal you didn't meet and how you handled it.What do you do if you disagree with someone at work?