Algorithms

Algorithms in practice

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Coding Interviews

- Many coding interviews are algorithmic-based. Only a small subset of algorithms occur frequently in interviews
- Sorting, Searching, Graph (DFS, BFS), Binary Search, Backtracking
  - Less common: Greedy, Dynamic Programming, Divide and Conquer
  - All data structure concepts are common in interviews
- Does this mean we shouldn’t study other algorithmic topics?
  - Definitely no. They have significant added value for both background and skills
  - Some of them may come in an interview, but rarely occur

Algorithmic-based projects

- Some projects are based on specific areas in algorithms
- For example, Google Maps must utilize a lot of graph algorithms to find the different routes from one location to another
  - From - To
  - The time
  - Transportation means: car, bus, train, etc
- Similar sub-areas exist in Twitter and Facebook to model relationships
- Cryptography utilizes concepts in number theory
- Some research projects utilize algorithms areas (Graph, Dynamic Programming, Greedy and the Approximations algorithm)

General Software Project

- In 95% of careers, you won't need to code an algorithm by yourself
  - This includes most project in big companies, such as Google and Facebook
- Some students take this point to decide to skip such a course
  - But you will find all strong engineers recommend being strong in DSA. Why?
  - **Analogy**: many of the exercises professional footballers perform in training are never directly used in real games, but they play a big role behind their skills!

Learn many things on a small scale!

- Tackling computation problems teaches you many concepts
- Thinking about correctness of what we code
- Enumerating test cases including critical boundary cases
- Sharpens your thinking, coding, debugging and testing skills
- Strong sense of the efficiency of what you code
- Acting like a problem-solver
- Tackling a problem from several angles. Thinking about the trade-offs

Algorithms and the CS core

- The CS community had a long journey building algorithms for many things
  - **Compiler**, interpreter, or assembler that are used behind the scenes eventually
  - Routing in **networks** relies heavily on algorithms
  - **Operating systems** scheduling algorithms
  - Clipping algorithms in **Graphics**
  - Encryption algorithms in **Cryptography**: (e.g. ElGamal encryption system)
  - Recovery algorithms in **Databases**
  - Pagerank algorithm in **Search Engines**
  - Hungarian algorithm in tracks matching in **Computer Vision**
  - Dynamic Programming for Markov decision process in **Machine Learning**

Questioning algorithms importance nowadays!

- In the past, no one questioned the importance of algorithms!
- Nowadays, as we build on very high level languages, frameworks and algorithmic libraries ⇒ some people question their importance
- You can build **basic** mobile app nowadays without many CS fundamentals!
- Solid computer science background is a fundamental key for a strong SWE
- During your journey you will certainly need to know the basics of algorithms!
- Tackling algorithmic challenges can be frustrating many times, but it is a game-changer for your thinking mentality!
    - It is also fun for people who like to be problem solvers!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."