Coding interview

- Algorithms and data structures
  - Come up with a way to accomplish the set task
  - Talk about its efficiency and trade-offs
  - Be able to change it as needed as the requirements change
- General coding ability: use a language you know well!
  - Write reasonably well-formatted code, good variable names, etc.
  - I don’t care about minor syntax errors or remembering the exact library APIs
- Be able to talk through the code
  - Suggest test data and step through the algorithm
  - Justify design decisions
  - If you’re stuck, talk it through. I can’t offer hints if I don’t know what you’re thinking.
- Edge conditions and error handling
- Maybe describe some tests if time permits

Design interview

- Thinking about large-scale design, possibly with no code written. More room to demonstrate your skills and creativity!
- Specific skills:
    - Breaking down problems into solvable parts (vs not subdividing)
    - Identifying & analyzing tradeoffs (vs no estimates & over-engineering)
    - Navigating different levels of abstraction (vs “rat holing”)
- Often underspecified; pulling requirements is crucial
    - “Should I optimize for … ?”
    - “Can I assume the input is … ?”
- Consider pseudocode as communication tool
- At Google, normally one of the five questions is a design question

How do you prep?

- Study an undergrad textbook on data structures and algorithms
- Know the basic data structures, paying particular attention to:
  - How to use them
  - Their runtime and memory efficiency (Big-O notation)
  - How to traverse them (trees, graphs)
- How the work internally (less important, but questions do come up)
- Find tech interview questions online and practice them!
- Get comfortable “writing code” on a whiteboard or piece of paper
- Practice full-scale interviews with a friend

- If you are interested in Google, feel free to contact me: jakehartman@gmail.com