Part 3

Interviewing

Common parts of an interview structure

Most interviews are hybrid: online + in person.

1. Phone screen

   Basic questions to check if you're a reasonable fit: background, algorithms

2. Behavioral interview

   Questions on past experience and hypothetical situations

3. Engineering interview

   Estimation, system design, tech quiz

4. Coding interview

   Algorithms on paper or coding

5. HR interview

   Typically short, towards end of circuit

Some examples from Switzerland (company redacted)

Each company does it a bit differently:

- Online screen
  + 3 x 45 minutes of behavioral, engineering, coding
- Phone screen
  + 4 x 30 minutes of coding and engineering
- Phone screen (rapid-fire)
  + 1h HR
  + 3h { engineering + code review }

General interview tips

- Be engaged and excited
  Know why you want to be there
  Have a pitch — interviewer may not have read your CV
- Make a good impression
  Be on time
  Dress adequately — in many cases this means no suit / tie
- Come in informed
  Read up on the company — culture, principles, practices… and interview process!
  Don’t ask uncomfortable / illegal questions
- Don’t waste opportunities
  Pick your questions carefully — ask strategic questions
- Don’t put yourself in a bad situation
  Don’t drink (too much) — get sparkling water
  Don’t ask for a smoke break
  Don’t eat dumb food — no lobster please
  Tell them ahead of time if you need accommodations

Part 3.1

Behavioral interviews

Behavioral interviews — **Examples**

- Tell me about a missed deadline and how you approached it.
- Have you had to deal with conflict with work colleagues? If so, how did you resolve it?
- What's the worst mistake you've made at work?
- What is your top-one achievement in your previous job?
- What's the most interesting bug you've ever fixed, and how did you approach it?

Behavioral interviews — **Technique: STAR**

Tell a story from your own experience

Situation:

Lay the groundwork for the story, describe an issue, give details

Task:

Describe what your responsibility was in that moment

Action:

Describe the action you took to move forward

Result:

Explain what results your action led to

Also applicable to CVs!

Behavioral interviews — **Tips**

- Stand out

  Have interesting stories — collect them over time!

  Don’t tell us about missed homework deadlines — please.

- Be memorable

  Chose a diverse set of examples. Don’t use the same job for every question.

- Be positive, enthusiastic, and humble

  Don’t complain or mock the questions!

  Don’t criticize your previous colleagues, workplace, etc.

- Be concrete

  Give numbers, facts, measurable outcomes

Behavioral interviews — **How to prepare**

- Collect stories

  Write down small summaries of interesting situations that you encounter.

- Seek responsibility

  Get involved at school and work.

- Practice the STAR method

  Can also be used on your CV

- Train with friends

  Use the questions given above

- Learn to think on your feet & react to unexpected situations

  Do improv

  Join a D&D group?

Exercise: STAR

- What is the hardest tech problem that you've had to deal with?
- Have you ever had to resolve a disagreement about a technical topic between two coworkers?

Part 3.2

Engineering interviews

Conceptual, Quick-fire, System-design

Conceptual engineering interviews

Intended to get you to think — precise answer irrelevant
Falling out of favor

- How many cow farmers are there in Appenzell?
- What has more calories? A smartphone battery or a cookie?
- How much would the Matterhorn weight if it were made entirely of Toblerone candy?
- What does it mean if the probability of rain tomorrow is 60%?

Break the problem down, make assumptions and guesses, explain your thinking, use back-of-the-envelope calculations and fermi estimates.

Quick-fire engineering interviews — **Examples**

Intended to gauge your scientific and engineering fundamentals

- How do you insert an element in a linked list?
- How do you check if a linked list has a cycle?
- What is a promise and what do I use it for?
- Why is it unsafe for generic arrays to be covariant?
- What is a type class and why would I prefer that over an interface?
- What is the point of unit tests if you already have integration tests?
- What is more efficient, a tail-recursive function or a loop?
- Why would you use exceptions over the `Result` monad?
- What is the problem with rebasing after pushing to upstream?
- What is the complexity of quicksort when using linked lists?
- What is the difference between `#:::` and `++` on lazy lists?

Quick-fire engineering interviews — **Technique**

- Don’t make things up

  It’s OK to say you don’t know.

  Offer something related if you can.

- Ask clarification as needed

  Typically should be needed often.

- Prepare ahead of time

  Review CS-214 debriefs, for example.

System design engineering interviews — **Examples**

Intended to gauge your real-world experience / streetwise

- Design a task-list app based on given framework (**webapp!**)
  - How many clients would this support?
  - How would you stress-test this?
  - What is missing to make it realistic / robust?
  - How would you distribute this across multiple servers to scale?
- Analyze the safety / longevity / maintainability of this particular design
- Design the architecture for a distributed key-value store with (some specific parameters)

System design engineering interviews — **More**

Sometimes about your previous experience instead:

- Have you ever developed a webapp?
  - Which framework did you use, if any?
  - How did you architect it (draw diagram)
  - How did you come up with this design?
  - How did you test it?
  - How long did it take to develop?
  - What bugs did you have and how could you have prevented them?

System design engineering interviews — **Technique**

- Communicate throughout
  Explain what you're doing at all times.
- Ask lots of questions
  Make sure you're designing for the customer.
  But don't fish for answers.
- Restate the problem as the interview evolves
  Confirm that you're on the same page.
- Make guesses / assumptions and check them
  Communicate with your interviewer.
- Offer alternatives
  Mentioning an option even if you don't explore it is still valuable.
  Clarify trade-offs and design choices.
- Draw and write
  Much easier on a whiteboard — use it!

System design and quick-fire — **How to prepare**

- Be reflective

  Think back after every grade, lab, completed project.
  Don’t settle for superficial understanding.
  Ask for feedback and give feedback.

- Learn from others

  Never be the smartest person in the room.
  Get involved in projects with experts.
  Read blogs.
  Program in pairs.

- Learn from the pros

  Read problem solutions, lab scaffolds, source code.
  Read and criticize Ed, StackOverflow questions.

- Train yourself at answering questions

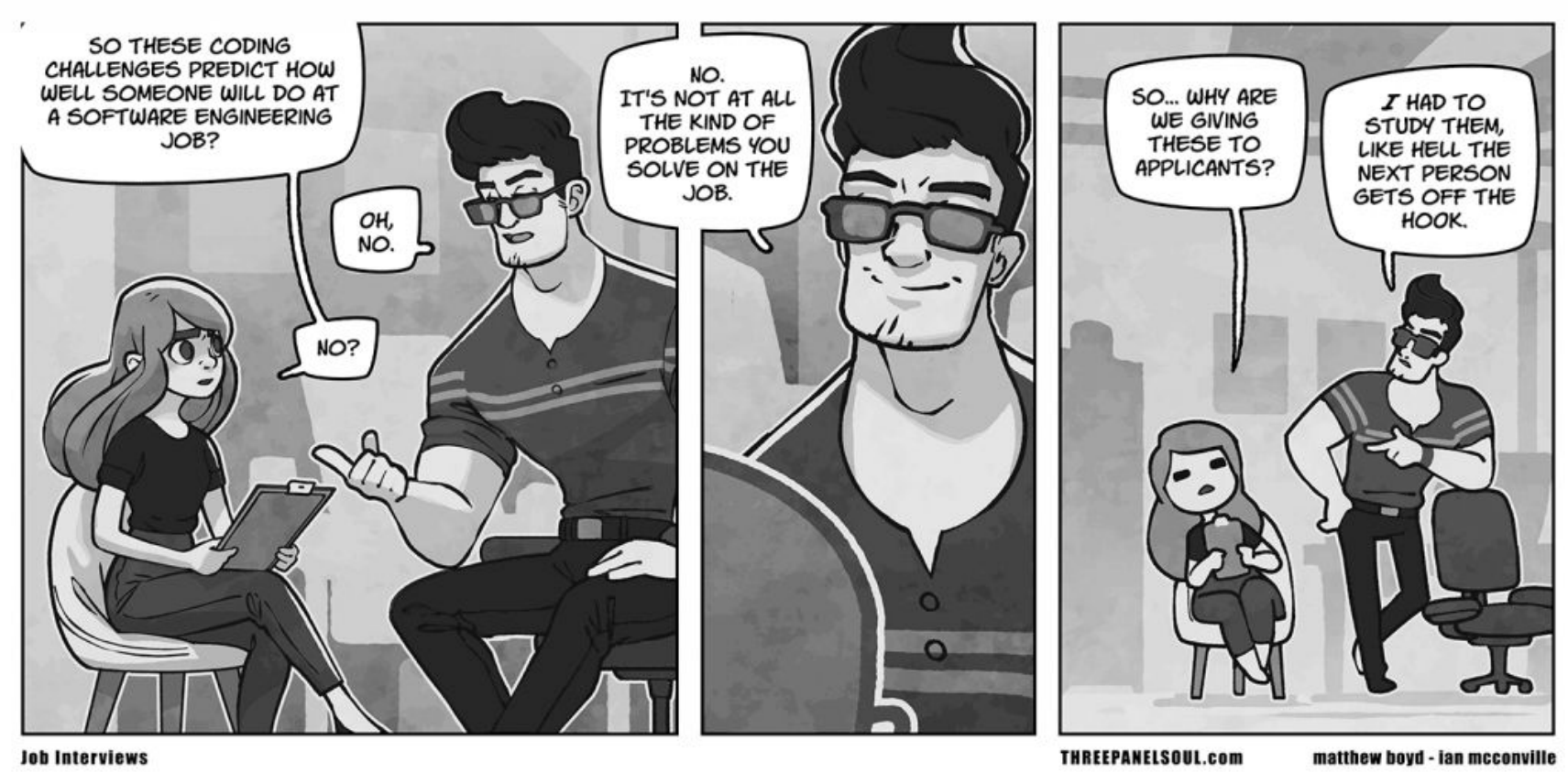
  Contribute on Ed, get involved in education
  Make money from it: private lessons for gymnase students, assistantships

Exercise (in exercise session)

Answer the quick-fire questions listed above

Part 3.3

Coding interviews



Coding interviews — **Intro**

- Typically 20 minutes to 1 hour
- Free or not to pick language
- Typically one of
  - Algorithms design
  - Programming puzzle
  - Code review (performance, correctness)
- Almost never in an IDE / executed

Examples: Almost all of our on-computer exercises + lots online

Coding interviews — **How to prepare**

- Pay attention in BA4 algorithms class
  Know the classics
- Use interview websites, books
  Practice, practice, practice
- Participate in programming challenges
  E.g.ICPC, advent of code
- Be reflective
  Give and seek feedback
- Sign up to be an assistant for a class
  Best way to get unexpected questions and have to think fast
- Network!
  Set up a question bank
  Use the Google drive!

Coding interviews — **Tips**

- Communicate, ask questions, speak while coding
  - Describe your thinking.
  - Speak while coding..
- Take the time to write correct code
  - Don’t jump right in!
  - Remember tests, docs, and specs
- Be comfortable with debugging
  - You’ll need to think fast when you’re told your code is wrong
- Get enough sleep
  - Matters enormously
- Don’t cheat, don’t lie
  - Reputations travel fast

Some useful resources

- Books
  - McDowell, *Cracking the coding interview*
  - Bentley, *Programming pearls*
  - Okasaki, *Purely functional data structures*
- Online resources
  - ETHZ CV guide
  - The Workplace Stack Exchange
  - Advent of code (Scala solutions), Project Euler
  - ICPC
- Parting words
  - Be generous to your peers
  - Help each other
  - Good luck!

Exercise (in exercise session)

Attempt a few interview puzzles

Demo

A coding interview

Thank you!