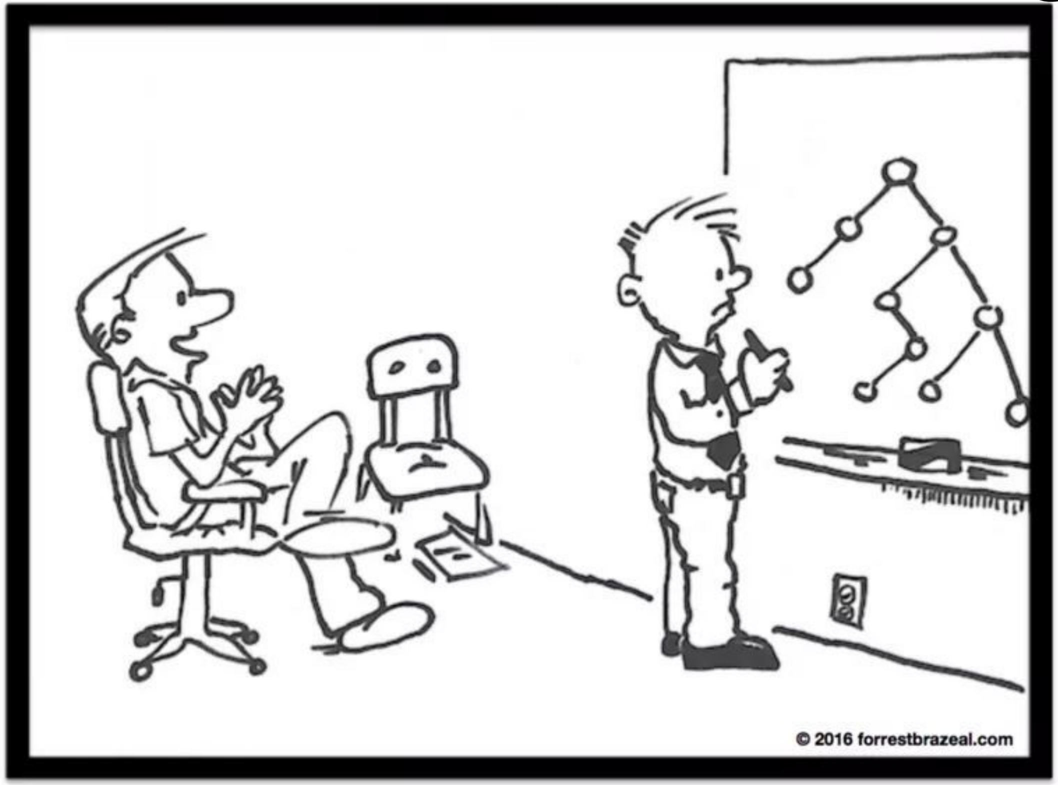
Technical Interviewing



"We want our interviewees to solve real-world problems. So while you balance this binary search tree, I'll be changing the requirements, imposing arbitrary deadlines and auditing you for regulatory compliance."

TECHNICAL INTERVIEWING SUCKS!

Putting the candidate through the same bullshit you went through

Useless

Whiteboard Interviews

O RLY?

The Practical Developer
@ThePracticalDev

Learning Outcomes

By the end of this session, you should be able to:

- List common types of technical interviews.
- An organized approach to coding interviews.

Technical interviewing
is a rite of passage
to the best jobs
at the best companies.

Technical Interviewing Disclaimers

Everyone hates it (but most still play the game).

It is a unique skill (not directly related to school or job work).

Most Senior Data Scientists and Engineers are **not professional interviewers**.

They can ask about anything;You'll have to answer in any medium.

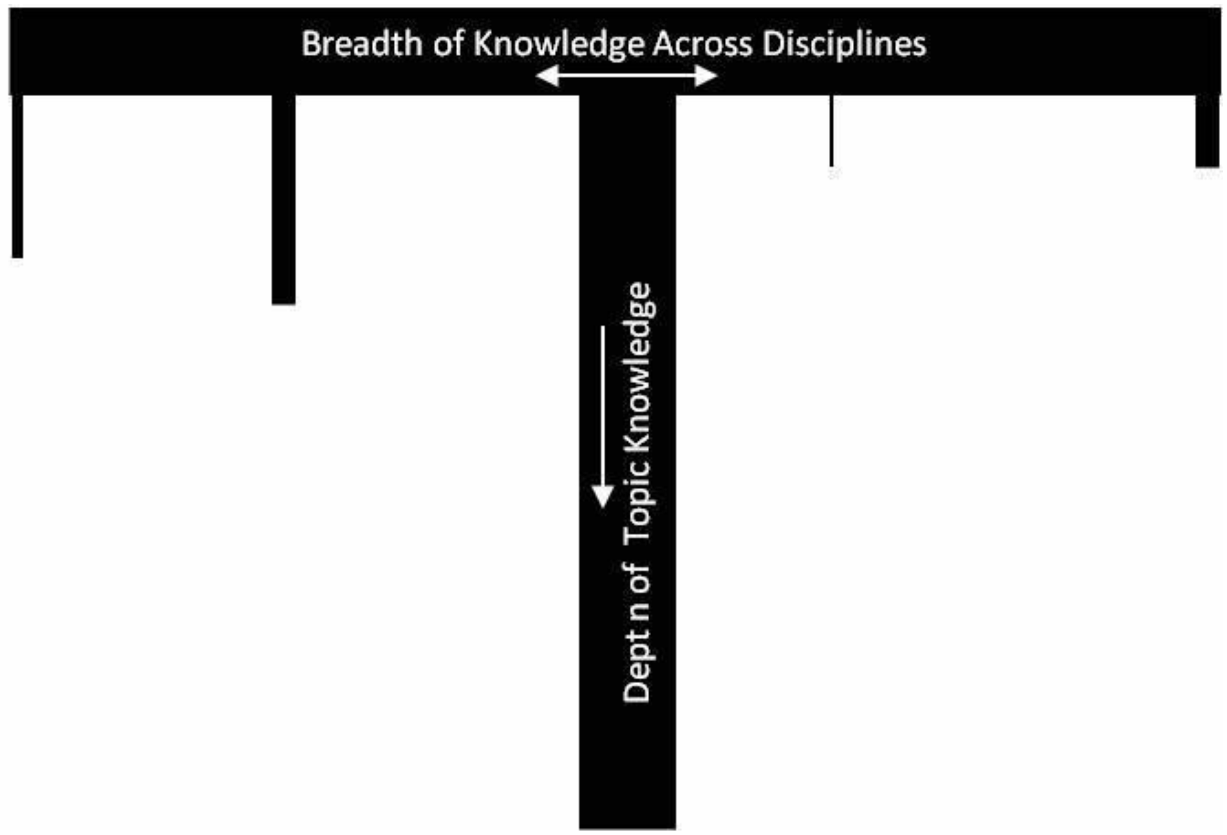
Technical Interviewing Question Types

1. General questions
2. Specific questions
3. Specific problems

General & Specific Questions

1. Can you talk about your knowledge using the standard jargon and in a logical manner?
2. Can you discuss ideas at different levels of abstraction? For different audiences?

”T” Model of Knowledge


Breadth of Knowledge Across Disciplines
Dept n of Topic Knowledge

General Questions Examples

1. What languages do you know? How would you rate your knowledge of each?
2. What is your favorite language and why?
3. Is more data always better?

Specific Questions Examples

1. I see you did a project with tf-idf. Explain tf-idf to me…
2. Why are Python dict’s ordered?
3. Why is it called “Logistic Regression” but it is really a classification technique?

Sources for Specific Questions

- Previous work
- School projects
- Your GitHub
- Their job posting
- The interviewer’s major, …

What should you do if you know the answer to a specific question?

- You have to be honest and say you don't have experience in that specific domain. Making up answers is failing an interview.

- Hopefully, you have adjacent knowledge

- One way to answer it, "I do not have hands-on experience in that. However since I saw this interview coming up, I have started my own research. Here are my ideas so far…"

Questions?

Specific Problems, aka Real Technical Interviewing

⅔ Result

⅓ Process

Student Activity

Solve on your own: You two variables that contain integers. You want to swap their respective values.

Not allowed:

- A temporary variable
- Multiple assignment
- Changing type to a container.

Brian's Method for Technical Interviewing

1. **A**sk
2. **U**nderstand
3. **P**lan
4. **C**ode
5. **T**est
6. **I**mprove

Ask Step

ALWAYS ask follow-up questions.

Almost all technical questions are underspecified.

Asking shows you are curious and do not make assumptions.

Understand Step

Create concrete examples of inputs & outputs.

Create the smallest example of the problem.

State the givens.

Ask for boundaries. For example, “Is it okay to use Python and its Standard Library?”

Plan Step

Brainstorm many options; then select most feasible.

Write comments first.

“Wishful programming” - create placeholder functions then fill them in.

Code Step

Note this is step 4

This is the goal but you should not rush into it.

Properties of High Quality Code

1. Runs - Always write working code
2. Correct - Meets spec (specification)
3. Logical - Well-ordered & Modular
4. Performant - No unnecessary looping
5. Written quickly - Be a fluent coder

Swap Values Solution

```
a = 42

b = -1

a = a +  b

b = a - b

a = a - b
```

Tactics

- Write comments as you go.
- Write a small amount code. Run that code. Write more code…
- Let Python do the work for you.

Test Step

Confirm that code works by stepping through the initial concrete examples.

Track state and final output.

Double check with common examples and edge cases.

Bonus points for writing actual unit tests.

Improve Step

Improve the code:

- Rename variables

- Erase unnecessary parts

- Improve performance

- Brainstorm about scalability and extensions

Swap Values -
Optimize with bitwise operators

```
a = a ^  b

b = a ^ b

a = a ^ b
```

XOR swap algorithm

Worked example

Code Review Rubric

Attitude Matters

- Did you give up? I probably don't want to hire someone that gives up a difficult problem.
- Did you talk through it? I probably don't want to hire someone that doesn't have a willingness to communication.
- If stuck, did you ask for help? I probably don't want to hire someone that won't ask for help.

Attitude Choices

- Always externalize your thought process:
  - Assumptions
  - Plan
  - Implementation
- Never quit!
- Check-in with interviewer - Try to make it collaborative.

Practice like how you will perform:

- Solve novel problems
- Do it on video
- Use online code editor or Google Doc

In Breakout Rooms

1. One person is the interviewer and the other person is the interviewee.
2. Both people can look at the problem https://gist.github.com/brianspiering/9562bf3dc63501e8194b5ef35d429608
3. Solve it in https://coderpad.io/launch-sandbox in incognito mode.
4. **Only the interviewer** should look at the solution https://gist.github.com/brianspiering/6e411af7b6b78f5fa24765586a2e95ae
5. Interviewer should give feedback at the end.

Additional Problem

Problem:

https://gist.github.com/brianspiering/ee99f83c02321d0eb7bf47c45aec0f31

Solution:

https://gist.github.com/brianspiering/dcb8cdcf1c45d23731d684d2e832
6f28

Bad programmers worry about the code. Good programmers worry about data structures and their relationships.

— *Linus Torvalds* —

Coding Interviewing Tips

Pick the Data Structures first. What should be your goto data structure?

Hash Map / Python’s dict

Know dict’s methods and variations.

Coding Interviewing Tips

After picking data structure, pick the algorithm(s). What should be your goto data algorithm?

Dynamic Programming / Caching - Reuse precomputed values.

Pro tip - Use a hashmap as your dynamic programming cache.

Brian’s Flashcards

Takeaways

- You should be fluent in explaining common technical concepts in Data Science.
- Apply **A.U.P.C.T.I.** in every technical interview.
- Practice, Practice, Practice.

Other Coding Interview Strategies

- Concrete examples (from specific to general)
- Nearest Neighbor (find a related program with known solution)
- Iterative refinement (brute force -> linear -> sublinear -> constant)

How to study for technical interviews

Studying for technical interviewing is fundamentally different from learning programming or studying for academic test (because they are fundamentally different than learning how to program or academic tests)

Your goal for a technical interview is to fluently type working solutions while explaining your logic. You should be memorizing common design patterns and be able to recognize their applications to novel circumstances.

How to study for technical interviews

I. Solve novel problems

- Try independently under actual circumstances. Time boxed
- Then compare solution to best possible solution that I can currently understand
- Make list of techniques to learn
- Create my own version of the solution
- Make flashcards for spaced repetition

II. Practice flashcards

III. Study new techniques

Pareto Efficiency for Algorithms

Memorize a single implementation for the following:

- Sorting - Quicksort
- Searching - Binary search
- Creating a Counter / defaultdict