Unit 2 Build Week Pt. 1

AGENDA

- Interview Practice Pitfalls & How to Avoid Them
- Binary Search
- *Lectures will be an hour long and focus on interviewing*

Interview Practice Pitfalls & How to Avoid Them

A LITTLE BACKGROUND ABOUT ME

- I was never naturally good at algorithmic interviews
- I failed a bunch of interviews and had impostor syndrome
- I lucked out on my intern interview and didn’t get asked data structures/algorithms

THINGS TO KNOW ABOUT PROGRAMMING INTERVIEWS

- For the most part, it’s a separate skill
- There are different types of interviews
- Data structures and algorithm interviews have a steep learning curve, but once you crack it then you will be able to ace a *good* amount of interviews

HOW YOU PRACTICE MATTERS

- How you practice has a huge impact on your performance
- Very few people practice effectively for interviews

"Most people have the will to win,
few have the will to prepare to win."
- Bobby Knight

COMMON MISTAKES WHEN PRACTICING FOR INTERVIEWS

- What do you think are the most common mistakes people make when practicing for interviews?

COMMON MISTAKES

1. Mindlessly doing as many problems as you can
2. Forgetting about problems/patterns already seen
3. Not being emotionally prepared
4. Getting demotivated
5. Not preparing for other types of interviews

MINDLESSLY DOING AS MANY PROBLEMS AS YOU CAN

- Amount of Leetcode questions done != Amount of your understanding of the material
- Brute-Memorizing solutions is not efficient
- Add ***deliberate practice*** to your regimen
  - "...a special type of practice that is purposeful and systematic"

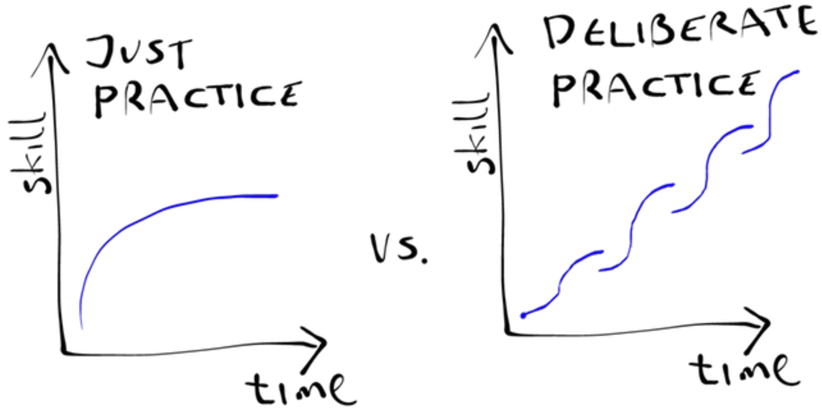
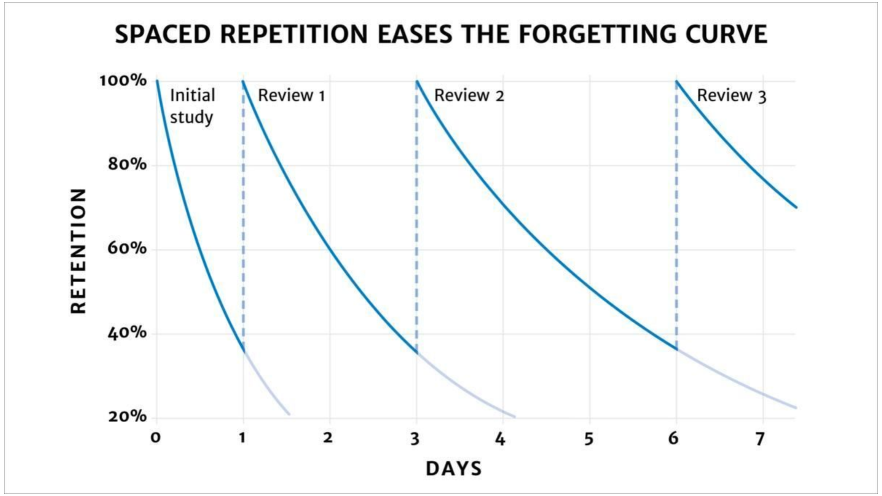


Photo from https://ideas.plusplus.co/mastery-through-deliberate-practice-827ef762fc0

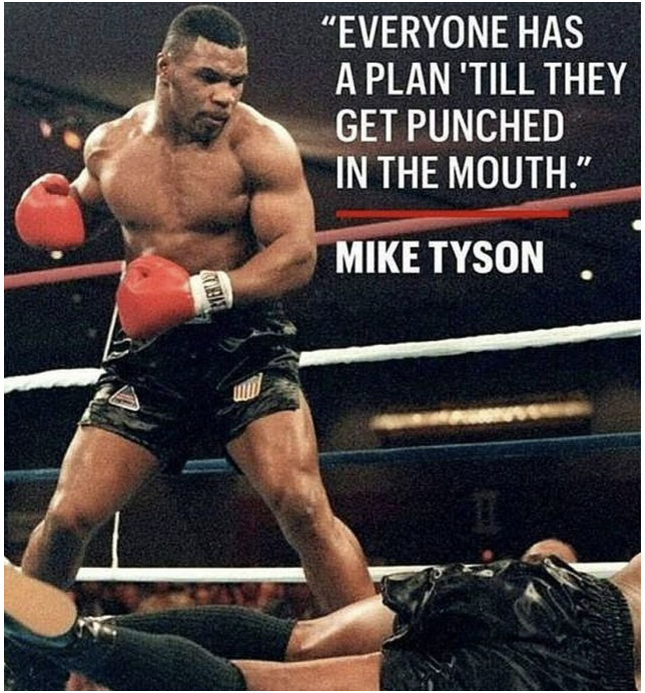
FORGETTING ABOUT PROBLEMS/PATTERNS ALREADY SEEN

- Acing interviews is largely on recognizing a pattern/solution you've previously used and applying it to the problem at hand
- Fast recall is important during interviews
- Add ***spaced repetition*** to your regimen to retain as much information as possible
  - Use *Anki* to create custom flashcards with built-in spaced repetition



NOT BEING EMOTIONALLY PREPARED

- The emotional and physical (pre-COVID) environment during interviews are different
- Simulate conditions similar to an actual interview:
  - time limit, no compiler, no autocomplete
  - practice interviews with others (e.g. Pramp)



GETTING DEMOTIVATED

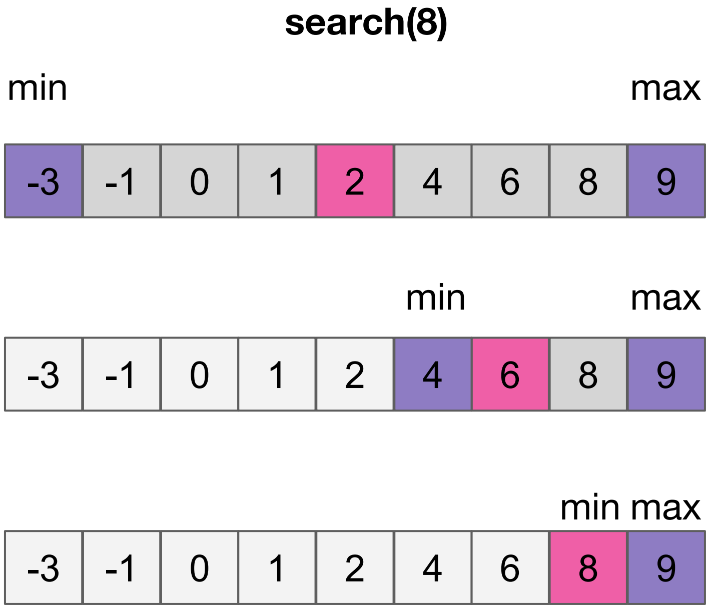
- Doing all hard problems is a recipe for demotivation, and ultimately failure
- Interleave easy-medium-hard problems together to gain confidence and challenge yourself
- Stuck in a problem? Look at the solution, then redo it later

"*A competitor needs to be process-oriented, always looking for stronger opponents to spur growth, **but it is also important to keep on winning enough to maintain confidence***" - *The Art of Learning: A Journey in the Pursuit of Excellence*

Binary Search

BINARY SEARCH

- Halve the search space on each iteration for **sorted** collections/ranges for O(logn) performance
- Take the middle element of the current search space
  - If that’s the target, then you’re done
  - If it’s greater than the target, go left
  - Else it’s less than the target, so go right

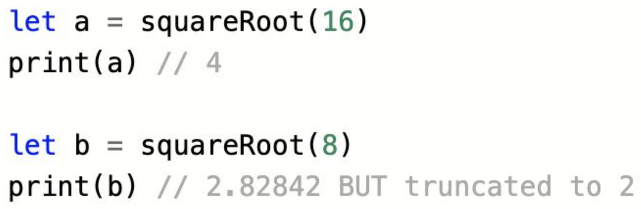

search(8)

BINARY SEARCH: THINGS TO KNOW

- Very useful for sorted collections (arrays, strings) but also ranges (range of values)
- Can be implemented iteratively/recursively
- **Keywords:** sorted, ranges
- **This should be muscle memory!**

BINARY SEARCH EXAMPLE: SQUARE ROOT

- Implement ***squareRoot(_ x: Int) -> Int***
- Find the square root of a number x, where x > 0
- Since the return type is an integer, the decimal digits are truncated and only the integer portion is returned



```
let a = squareRoot(16)
print(a) // 4

let b = squareRoot(8)
print(b) // 2.82842 BUT truncated to 2
```

SQUARE ROOT BRUTE-FORCE

- Try every value from [0, target] until we find the square root
- O(n) performance, where n = target

SQUARE ROOT BINARY SEARCH

- Use binary search to keep halving the search space
- **Key idea:** binary search also works for ranges of values, not just sorted arrays/strings
- O(logn) performance
- Leetcode Link

squareRoot(16)

search space/range: [0, 16]

0, 1, 2, 3, …. 8, 9, 10, ... 16

Square Root Binary Search Demo