The Key to Effective Studying

bytebybyte

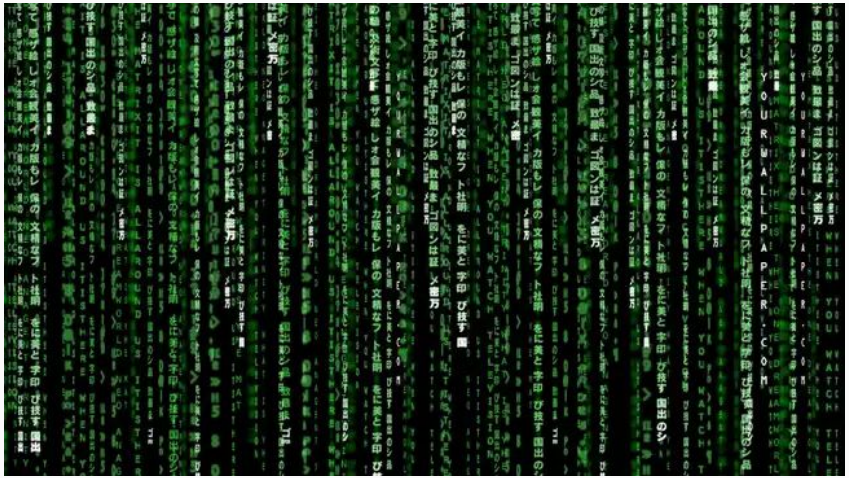
How to solve problems effectively

- What are the goals of practicing?
  - Simulate real-world interview
  - Identify and improve weaknesses
  - Become comfortable with the tools and systems
- Avoid
  - Rehashing what you're already good at

bytebybyte

Simulate real-world interview

- Hand write code
- Talk out loud as you go
- Test your code by hand
- Apply the strategies we will discuss today



byte by byte

Identify your weaknesses

- If you get the solution
  - Where (if anywhere) did I get stuck?
  - Copy code verbatim into compiler and test more thoroughly
  - Keep a list of mistakes you made. Next time, avoid those mistakes

- If you didn't get the solution
  - What part of the solution did you miss?
  - Break it down. What was the pattern (or patterns) you missed?
  - What topics were you weak on?

bytebybyte

Become comfortable with the tools and systems

- Interviewing is not like the normal job
  - Whiteboards
  - No Google
  - No compiler
- To be comfortable in the interview, you need practice in that environment
- You're also learning new strategies that you want to be able to use effectively

bytebybyte

“I don’t understand the solution…”

- This is totally okay, but you need to be able to figure this out on your own
- “What part don’t I understand?”
  - Alternatively, which parts *do* you understand?
- Walk through the tricky parts to see what’s happening
  - https://youtu.be/B3U6LExgevE
- Are you missing any knowledge that would help you understand the problem?
  - Keep a list of these topics so you know what to review
- Research the problem

bytebybyte

“I don’t understand the solution…”

```
private static void permutations(int[] items, int i, List<Integer> path,
                                 List<List<Integer>> results) {
    if (i == items.length) {
        results.add(new LinkedList<Integer>(path));
        return;
    }

    for (int j = i; j < items.length; j++) {
        swap(items, i, j);
        path.add(items[i]);
        permutations(items, i+1, path, results);
        swap(items, i, j);
        path.remove(path.size() - 1);
    }
}

private static void swap(int[] items, int i, int j) {
    int temp = items[i];
    items[i] = items[j];
    items[j] = temp;
}
```

Misc Advice

- How much time should you spend on a problem?
  - No more than 30 minutes unless you're making progress
- If you're short on time, you can usually skip the coding part
  - Most people are relatively good at the coding and struggle with problem solving
- How difficult should problems be?
  - Ideally not impossible but challenging
  - Easy problems are a good warmup to build confidence before interview

Your Problem Solving Framework

Why do we need a framework?

- Allows us to manage our time as effectively as possible
- Allows us to stay focused
- Helps us get back on track if we get lost
- Allows us to see where our weaknesses are to focus our studying

The Framework

1. Understand the problem [3-5 minutes]
2. Find a brute force solution [5 minutes]
3. Optimize your solution [15 minutes]
4. Code your solution [15 minutes]
5. Test your solution [5 minutes]

bytebybyte

1. Understand the Problem

- Make sure that you really know what is being asked. Take your time here.
- Write out the function signature. What are the inputs and outputs?
- Go through the example inputs and outputs. How does the input get that output?

Examples

- Print reverse linked list
  - Return void
  - Just print out all the values
  - `void printReverse(Node list)`
- 0-1 Knapsack
  - Each item can be only included once
  - What are we returning? A list of items
  - `List<Item> knapsack(Item[] items)`

2. Find a Brute Force solution

- Just find the dumbest/most obvious solution
- Ignore everything you might already know about the problem
- Some strategies:
  - Can you enumerate all the possibilities?
  - How would you solve the problem by hand?
  - Can you solve a similar/simpler problem?
  - Can you solve the problem in multiple stages?

Examples

- Print reverse linked list
  - Lots of different approaches here
  - Add all the items to another data structure
  - Iterate to specific indices
    - Implement a “get item at index” function
- 0-1 Knapsack
  - There is a finite number of different combinations, so we can just find all the different possible combinations

3. Optimize your Solution

- What is the time and space complexity of our existing solution?
- What is the Best Conceivable Runtime?
- Now we can use our existing knowledge
- BUD optimization
- We will cover this in depth in Module 3

Examples

- Print reverse linked list
  - Currently `O(n)` time and `O(n)` space
  - Can't do better with time but can we do better with space?
  - What if we modify the list itself?

- 0-1 Knapsack
  - Time and space are exponential
  - We are computing all these invalid combinations and storing a lot of extra data
  - We are also repeating computations
  - Dynamic Programming

3b. Explicitly Understand How to Code

- You should NOT be figuring out how to code stuff up while actually coding
- When you get to coding, it should just be “translating” the work you’ve already done
- Pseudocode is optional. Just make sure you understand how to implement the tricky parts

Examples

- Print reverse linked list
  - Literally sketch out exact steps to reverse the list

- 0-1 Knapsack
  - Pseudocode might be helpful for understanding recursion
  - Let's lay out specific functions we need
  - If we're using tabulation, what does that array look like and how are we populating each cell?

4. Code

- This part should be easy if you did 3b
- Go as fast as you can here while being accurate
- When practicing, do this by hand and then test it with the compiler when you’re done

5. Test your code

- ALWAYS test your code
- Try to walk through the code as if you were the compiler
- Don’t assume that the code does what you intended it to do
- You should be able to confidently say your code is correct
- https://youtu.be/HgthIgvdMJ8

Tips and Tricks

- Follow the steps in order
- Make sure you do them all
  - I recommend practicing with a checklist
- This week's exercises are designed to target each step of the framework, so follow along

bytebybyte

Finding a Brute Force Solution

https://leetcode.com/problemset/all/