Class six

Summary

- Continuation of the exploration of the Articulation Point problem, with side discussions on slowing down, creativity, and handling the discomfort of not knowing.
- Summary. Quick reminder about midterm and the reflection essay. (250-500 words) - will be collected on October 18th. We will answer any questions you have about interviewing and talk about the behavioral part of the interview.

Practice

- Summary of the various practices we have covered
    - Creativity and assimilation practices
    - Productivity and energy management practices
    - Communication and confidence practices

Class five

Summary

- Confidence framed as a sense of comfort in who you are and what you know (and not know) at any given moment.
- Assertiveness: taking ownership of your words. Too many explanations and filler words cloud the key points of your message.
- We started with the lovely Articulation Point problem in a directed acyclic graph. We are using this problem in this class and next.

Practice

- Check in with yourself about how comfortable you are feeling with yourself at any given moment. Start small (noticing in more neutral moments - where you are likely to feel ok about yourself).
- Start practicing clarity and assertiveness in communications: emails, chats, (and spoken).
- Reminder to revisit the earlier practices of “slowing down” and “comfort with discomfort”.

Class four

Summary

Post work: Watch this 30-min video on an array problem (Learnings from this will transfer over to a few more problems when you prepare for interviews)
We covered a lot of stuff in this class! We looked at the idea of Contract/Story/Details

- We applied this idea to understanding the key characteristics of a data structure, an algorithm, an email!

We finished our coding fluency exploration of the Inflection point problem.
Lifecycle of a problem:

1. Understand the problem statement [clarifying questions, good examples]
2. Play with the problem (Optional: Highly recommend that you try this every week a few times. If we spend time with the problem, the solution we learn or discover later on gets assimilated a lot better)
3. Bruteforce solution and runtime
4. Optimal approach idea
5. Look at the solution (Optional: this could be a quick peek - equivalent of a hint, or looking at the entire solution and understanding it really well.)
6. Describe the optimal idea (in an interview - make sure you have agreement on the approach before you start coding it)
7. Code!
    a. Coding Fluency! Make sure you understand the code well - do some coding fluency explorations around base case, loop conditionals etc [like we did in class]
8. Testing when applicable
    a. As you make all test cases pass, pay close attention to the changes you make to the code. Make sure you understand why that change fixes the test case.
9. Reflection (Optional: What surprised you the most? What was the most fun? Most annoying? Did any extensions of this problem crop up in your mind?)

Practice

- Use the 9 steps to support and serve your preparation. These are guiding principles, not written in stone.

- When preparing for interviews, track your progress with problems, so you can repeat problems (every 10 days or so) so the underlying learning from that problem goes deeper.
- Try problems in Breadth First Order, don't spend weeks on one topic.
- When working on a problem, sometimes work it out on paper and sometimes use an IDE.

Class three:

Summary

- Inflection point problem!
    - Lifecycle of a problem
- Coding Fluency
    - We looked at the bruteforce solution and the optimal solution.
    - It is important to know the time and space complexity of using a subroutine of Python that splices an array - does it create a new array? As an engineer, you need to know these fundamentals of your chosen programming language.
    - We looked at the optimal solution and started to understand the intention behind each variable/line to understand the nuances of well written code. We will continue this next time.

Practice

- Coding Fluency!

Class two:

Summary

We continued the exploration of the Local Minima problem further, we wrote the code for it and took the code apart a bit to understand the skill of coding fluency. We also While we had fun with this problem:

- We discussed the importance of how we relate to discomfort and setbacks and how that has an impact on our productivity and creativity.
    - The transition from "I feel bad" to "Something is wrong"
- We discussed the role of curiosity in creative problem solving.

Practice

- Curiosity journal: Keep a notebook and write down any questions that come up for you in meetings/conversations, or while reading code or documents. They don't all need to be answered, we are trying to get good at asking them. With time, we can hone the skill of discerning the effective questions that dismantle unfounded assumptions and generate new approaches.
- Comfort/Friendship with Discomfort: Noticing when your energy is down due to a setback, no matter how minor, pausing and labeling it as a moment of discomfort. Reminding yourself that it belongs and that it does not mean anything about who you are and what can happen for you.

Class one:

Summary

We discussed the following ideas

- survival brain vs thinking brain
- Working memory and long term memory and their communication channel
- What typically happens when we work on a difficult problem and it starts to feel like a test.
- We brainstormed the "Find a local minima in a Binary Tree" problem and played with it with friendship and curiosity before attempting to solve it.

Practice

When you notice that you are faced with a difficult or unfamiliar problem and you sense tension/tightening, try the following practices:
(So, noticing that you are in that situation is an important part of this.)

- Slowing down and pausing
    - Breathing gently
    - Writing slowly or with non dominant hand (Geeta's favorite technique)
- Set the intention to befriend the problem - Dance not fight

- Working in timed sessions - set a timer for say 15-20 minutes, and when the timer rings reflect how that time went, if you were dancing vs fighting - and then repeat.