DSA Learning Plan and strategy

1️⃣ **Stick to the Roadmap** : [DSA Roadmap]

2️⃣ **Master the Basics First** –

- Start with fundamental data structures like **string, array, stack, and queue** before moving to advanced topics.
- Focus on **string and array mastery** before solving any question.

3️⃣ **Deep Dive into Each Data Structure**:

- Understand the Data structure concepts.(Follow any playlist on YT)
- Implement all its operations (insert, delete, reverse, etc.) by checking out the solution.
- After this try to **implement them by yourself**.
- Operations are the key. If you are able to manipulate operations, you will be able to solve questions effectively later on.

4️⃣ **Maintain a Detailed Notebook** 📒

- Note down everything you learn about that Data structure and different **patterns** you observe in problems.
- Write key operations, variations, and approaches for each topic.

5️⃣ **Find & Solve Pattern-Based Problems**

- Search the particular topic on geeksforgeeks or other sites and find out top 5-10 questions covering different patterns.
- Write them down and **analyze patterns before solving**.
- After that try to solve a similar problem by yourself.

6️⃣ **Create a github Repository**

- Create a github Repo.
- Create a folder for a DS and subfolder for operations, easy, medium problems and even for patterns.
- Whatever the code you are solving on leetcode or any platform, create a class in the repo and add the working code with adding a commented question.
- This should be your own written code.
- Add comments wherever you feel it is needed.
- Revise concepts from this repo by looking at your own written code.
- **THIS WILL BE THE BEST WAY TO REVISE.**

HOW TO APPROACH QUESTIONS?

7️⃣ Try Before Seeking Help

- Attempt the problem on your own, even if the solution is inefficient.
- Don't think that you are writing ugly solution.
- Forget about time complexity and write the brute force solution even if this is wrong.
- **Do not check solutions immediately.**

Process of solving:

1. **If you can't even write a solution** → Ask ChatGPT/AI tools for an algorithm. Try converting this algorithm into code.
2. **If you wrote something but it's incorrect** → Ask ChatGPT to check where you're wrong.
3. **Once you get the brute-force approach** → Implement it, then analyze how to improve time complexity.
4. Ask ChatGPT for ways to improve time complexity and create a better solution.
5. **NEVER EVER LOOK THE PERFECT SOLUTION BEFORE TRYING BY YOURSELF. NEVER.**

8️⃣ Optimize Your Code

- After implementation, compare with the best solutions.
- Improve **time complexity and approach** by analyzing optimized methods.

9️⃣ Calculate time and space complexity

- For every solution you are writing, try to calculate time and space complexity on your own and note it down.
- By doing this your skills of calculating time and space complexity will improve.

🔟 Master Each Topic Before Moving On

- Once comfortable, create **short notes** summarizing all patterns for quick revision.
- Use this sample short notes sheet to format your short notes:
    - How to create own short Notes

***Directly looking for the solution without approaching the problem is a bad habit that can't even be fixed in 21 days!* – NISHCHAL**

FOLLOW @codewithnishchal on IG