Fastest Way to Learn DSA

Intro

Are you struggling with DSA (Data Structures & Algorithms)?
Do you open LeetCode and feel lost within 30 minutes?
Do you feel like you've wasted time trying random DSA plans that never work?

Let me tell you something clearly —
DSA is **not** about solving 500 problems blindly.
It's **not** about following some fancy "100 Days Plan".

If you want to master DSA **fast and effectively**, you only need three things:

1. **Right Direction**
2. **Powerful Strategy**
3. **Strong Mindset**

Without these, you're just wasting your time and energy.

Let's begin the **smartest and most effective way to learn DSA.**

About Me (Why Should You Listen?)

I'm a Software Engineer at Zoho.
Here's my DSA background:

- 1200+ problems on LeetCode (Knight badge)
- 600+ on GeeksForGeeks
- 2500+ total problems across all platforms like Codeforces, CodeChef (4-star), HackerRank

It took me **4 years** of hard work to learn what I know today.

Now I'll show you how to do it **faster** and **better.**

The Mindset: DSA Builds Your Thinking Power

DSA is **not** just a technical subject.
It's about how you **think, break down problems, and solve them logically.**

To learn it fast, you need to choose a learning style that fits you. There are **3 types of learners:**

1. Curiosity-Based Learner (Explorer Mode)

These are people who enjoy solving problems for fun.

- They pick a problem and try to solve it with whatever they know.
- If they get stuck, they explore the tags/topics related to the problem (like Binary Search, DP, etc.).
- Then they learn that algorithm, understand how it works, and return to solve it.

Why this works:

- You're not memorizing — you're building real understanding.
- You explore multiple ways to solve the same problem.
- You learn where your current knowledge fails.
- You think deeply before seeing solutions.

This style is slow in the beginning but builds **strong problem-solving skills** for the long term. If you have more time to learn dsa.

2. Structured Plan Learner (Systematic Mode)

These people love having a plan and follow it step-by-step.

Here's how it works:

- Pick a topic (e.g., Binary Search, Recursion, DP, etc.).
- Solve:
    - **5 Easy problems** with a high acceptance rate
    - **7-8 Medium problems** with decent acceptance
    - **5-6 Hard problems** for advanced practice

This way:

- You fully understand the algorithm.
- You master each topic properly.
- Your confidence grows step by step.

Use this link for dsa learning order : DSA Learning order

This is great for people who want a clean path to follow.

3. Hybrid Method (Best of Both Worlds)

This is the method I personally use and recommend.

- First, try solving a problem using your logic.
- If you can't, go learn the algorithm behind it.
- Then return to the problem and solve 5 Easy and 3 Medium problems using that algorithm.
- Keep repeating this pattern.

This method:

- Gives you direction and discipline
- Builds mindset and thinking power
- Is faster than Explorer mode, but more effective than only planning

Pro Tip: Maintain a list or notebook of what you're learning and solving.

Common Mistake #1: Not Dry Running Code

One big mistake I see — people write code and run it directly.

Please dry run your code first.

- Take sample inputs.
- Think step by step: what will the code do at each step?
- This builds your debugging skill and problem understanding.

It's tough at first, but it's a **game changer**. Your thinking becomes clearer and your mistakes reduce.

Common Mistake #2: Not Learning from Top Submissions

After solving or failing a problem:

- Go to the **"Top Submissions"** tab.
- See how experienced coders write cleaner, shorter, or more optimized code.
- Learn from their style and techniques.
- Ask: Why didn't I think this way? What did they do differently?

This will teach you:

- Coding standards
- Optimization techniques
- Better problem-solving perspectives

Feeling Stuck or Frustrated?

It happens to everyone. Don't quit.

If you're stuck:

- Take a short break
- Go for a walk or talk to a friend
- Come back with a fresh mind

Then:

- If still stuck, read the solution — but **don’t copy**
- Understand it fully
- Add the question to a “retry list”
- Come back after a week and try again **without looking at the solution**

This builds **long-term memory and true learning**.

Time Management: How Long Should You Spend on One Problem?

Max time: **1 hour 30 minutes**

- Try 30 mins with full focus
- If stuck, take a break (20 minutes)
- Try again with a fresh mind (30 minutes)
- If still stuck, see the solution , learn and add it to retry list (10 minutes)

Don’t waste hours on one problem. Move smartly.

Participate in Contests

Contests help you:

- Think fast under time pressure
- Practice with a timer
- Compete with others

This improves your performance and builds confidence for coding interviews.