



For DSA

100 hours of DSA, *in order.*

15 patterns, 300 problems, 10 weeks.
Ready before your first online test.

Made with an engineer who **works at**
Atlassian and teaches DSA at NYC.





Before you follow a plan, you should know who made it.

**Made with an engineer
who works at Atlassian
*and teaches DSA at NYC.***



Akshay

Senior Software Engineer, Atlassian

Worked on UPI payments at Jio Platforms. Teaches DSA at NYC.

He speaks for himself, not for Atlassian.



Our students are already *doing this plan.*

A DSA track with an engineer from Atlassian.
Practice runs without AI.



THE DSA
TRACK

One pattern at a time

EVERY WEEK

Practice sets without AI

BEFORE
INTERVIEWS

Timed tests and mock interviews



Where our students *work* *now.*

Students taught by the team behind NYC, at Sheryians Coding School. 1,200+ are placed.

HIGHEST PACKAGE



Kushagra Patidar

SDE 1, Amazon

₹47 LPA

[LinkedIn](#) ↗



Anuj Parashar

SDE, Care Health Insurance

₹22 LPA



Rishika Jain

Cisco

₹17 LPA



Mansi Patel

Airtel

₹15 LPA



Now, let's start *the plan.*

Eight short sections. Read them in order.

01 Why you solve problems in order

02 Your 10-week plan

03 The 15 patterns

04 One hour a day

05 Timed mock tests

06 Big-O, in plain words

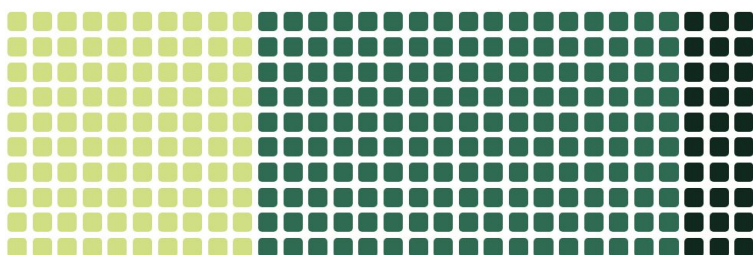
07 7 mistakes to avoid

08 After week 10

Why you solve 300 problems *in order*.

1,000 random problems teach you 1,000 problems.
300 problems in pattern order teach you to solve the 301st. A pattern is one method that solves a whole family of problems.

300 problems in 10 weeks.
Each square is one problem.



■ Easy 100 ■ Medium 170 ■ Hard 30

10 hours a week

10 weeks

1 language. Do not switch mid-way.

3 rules for the whole plan

1 One topic a week 25–40 problems

Solve problems of the same type until you spot the type before you finish reading the question.

2 25 minutes, then look Approach only

Stuck past 25 minutes? Read the approach, not the code. Close it. Then write the code yourself.

3 Keep a mistakes notebook Reread weekly

For every wrong answer, write what you tried, why it failed and the idea you missed.

Your 10-week *plan*.

10 hours a week. Each week builds on the last, so skip nothing.

1

Arrays and strings

Two pointers, sliding window, prefix sums.

40 problems

2

Hashing and sorting

Frequency maps, sort and scan, binary search.

35 problems

3

Linked lists and stacks

Reversal, fast and slow pointers, monotonic stack.

30 problems

4

Recursion and backtracking

Subsets, permutations, N-queens.

25 problems

5

Trees

Traversals, BST, lowest common ancestor, diameter.

35 problems

6

Mock week

Three timed mocks. Revise your mistakes from weeks 1–5.

7

Graphs

BFS, DFS, topological sort, union-find.

35 problems

8

Heaps, intervals, greedy

Top-K, merge intervals, scheduling.

30 problems

9–10

Dynamic programming

1D, 2D, knapsack, strings, tries.

70 problems and 3 mocks

By the end of week 10

300

problems, in order

15

patterns

6

timed mocks

The 15 patterns *coding tests use most.*

Learn to spot these. Once you see the pattern, you know how to start.

1 Two pointers

2 Sliding window

3 Prefix sums

4 Hashing / frequency

5 Binary search on the answer

6 Fast and slow pointers

7 Monotonic stack

8 Tree traversal + LCA

9 BFS / DFS on grids

10 Topological sort

11 Union-find

12 Heaps: top-K, merge K

13 Intervals

14 1D / 2D DP

15 Backtracking

For each pattern, in this order

1

template

3

easy

5

medium

1

hard

Then keep practising the same pattern until the week's count is done.



At NYC: DSA runs as a track with Akshay, an engineer from Atlassian. It goes one pattern at a time, with practice sets for each one.

One hour a day, *in the same order.*

One session

60 minutes

0

10 min · Reread yesterday's mistake

Solve it again from memory. If you cannot, it goes back in the queue.

15 min

35 min · New problems, timer on

Start with brute force: the simple, slow answer. Say its complexity, then improve it.

30 min

45 min

10 min · Read the editorial

The editorial is the official solution. Read it even if you solved the problem. There is usually a cleaner way.

60 min

5 min · Notebook

Pattern, trick, what you missed.
One line each.

Your daily targets

25

minutes at most
before you look at the
approach

2

problems a day on
weekdays

5

problems on a
weekend day, plus
one mock

Timed mocks: *practise the real test.*

Online coding tests last 60–90 minutes, with 2–3 problems and no other tabs. Your mocks in week 6 and weeks 9–10 copy that.

● ● ●

Your mock test

🕒 90:00

1

Problem 1

Medium

2

Problem 2

Medium

3

Problem 3

Hard

× No AI

× No search

× No autocomplete

Use a plain editor. Stop at 90 minutes, no extra time.
Submit even if your answer is partial, because partial marks exist.

After every mock: the review is the point

- 1 Find the pattern you did not recognise.
- 2 Check where the time went: reading, coding or debugging.
- 3 Next morning, rewrite the one you failed, cleanly.
- 4 Add the gap to your plan for the week.

6

mocks across the 10 weeks

12

mocks before your first real test

70%

solved in time means you are ready for interviews



At NYC: timed tests run without AI, like the online coding test. Mock interviews are run by working engineers.

Big-O, *in plain words*.

Big-O tells you how much slower your code gets as the input grows. Here n is the size of the input, such as the length of an array.

FASTEST AT THE TOP

BAR: HOW FAST THE WORK GROWS

 $O(1)$

Hash lookup

Map get and set, array index, stack push and pop.

 $O(\log n)$

Binary search

Sorted array, heap push and pop, balanced BST.

 $O(n)$

One pass

Linear scan, BFS or DFS on n nodes, two pointers.

 $O(n \log n)$

Sort

Sorting, a heap over n items, divide and conquer.

 $O(n^2)$

Nested loops

Pairs, simple DP, bubble sort. Expect the question: can you do better?

 $O(2^n)$

Subsets

Backtracking that tries every option. Only works when n is 20 or less.



In an interview

Say the brute force complexity, then the improved one, before you write any code.

7 mistakes that *slow down* your practice.

1. ~~Watching solutions~~

 Watching is not solving. Start a timer, open a blank editor, then compare.

2. ~~Random problem order~~

 Patterns build on patterns. Do trees before graphs, and recursion before DP.

3. ~~Switching languages~~

 Pick one in week 1. Stay with it for the whole 100 hours.

4. ~~Never timing yourself~~

 The test has a clock, so your practice needs one too.

5. ~~Skipping the notebook~~

 Without it, you repeat the same mistake in week 9.

6. ~~Hard problems too early~~

 Hard problems in week 2 only make you afraid of them. In week 9, they build confidence.

7. ~~Only DSA, no projects~~

 Interviews ask about your project too. Keep one project going while you practise.

After week 10: *keep practising.*

Every month after the plan

WEEK 1

3 problems
1 mock

WEEK 2

3 problems
1 mock

WEEK 3

3 problems
1 mock

WEEK 4

3 problems
1 mock

MONTHLY

1 contest

EVERY WEEK

3 problems and 1 mock

Mix the patterns, so you keep seeing every type.

EVERY MONTH

One contest

Codeforces Div 3 or the LeetCode weekly contest. Your rating is a side effect.

BEFORE A TEST

Reread your notebook

Two days with your notebook beat two days of new problems.

Where to practise. Pick one.

1 LeetCode

Best explanations and company tags

2 Codeforces

Speed and contests

3 GeeksforGeeks

Indian company patterns

4 NeetCode 150

A good ordered list, if you want one

The goal is not a number like “500 solved”.

It is spotting the pattern in 30 seconds, then explaining your approach out loud while you write clean code.

We teach DSA pattern by pattern, *live*.

● ● ● nyc · course

Covers this plan



WE TEACH THIS

Software Engineering + AI

Module 4 of Software Engineering + AI is DSA and system design, with weekly timed mocks in internship preparation.

 Live classes Engineer reviews For: **Software engineer intern (SDE)**

Talk to an advisor

See the course → notyourcollege.com



Guide by the NYC team, 2026. Problem counts are targets, not rules. Platforms named are independent of NYC. NYC does not guarantee employment or selection.