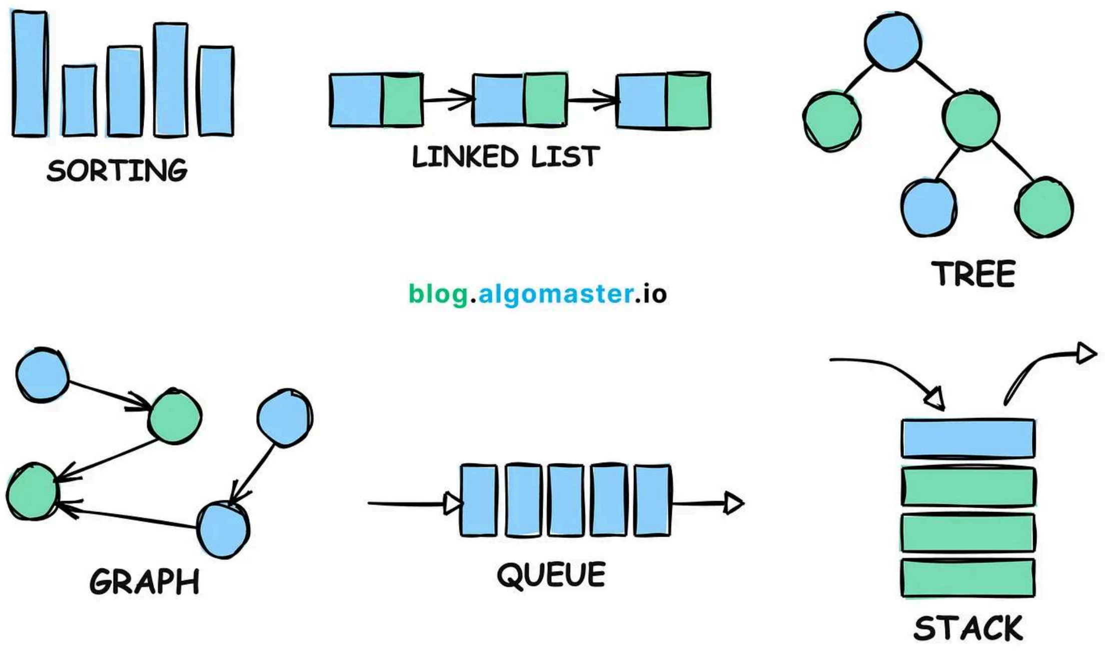
How I Mastered Data Structures and Algorithms

#16 How I mastered DSA

ASHISH PRATAP SINGH
JUN 16, 2024

620 21 47 Share



Getting good at **Data Structures and Algorithms** (**DSA**) helped me clear interviews at **Amazon, Google and Microsoft.**

But, mastering DSA was not an easy task.

I tried everything from **books**, **courses** (both free and paid ones), and spent thousands of hours solving **LeetCode** problems.

Looking back, I realized I spent a lot of time on things that didn't really help me get better at DSA. Had I focused on the right things, my journey would have been a lot easier.

In this article, I'll share tips that will help you master DSA more efficiently without feeling overwhelmed.

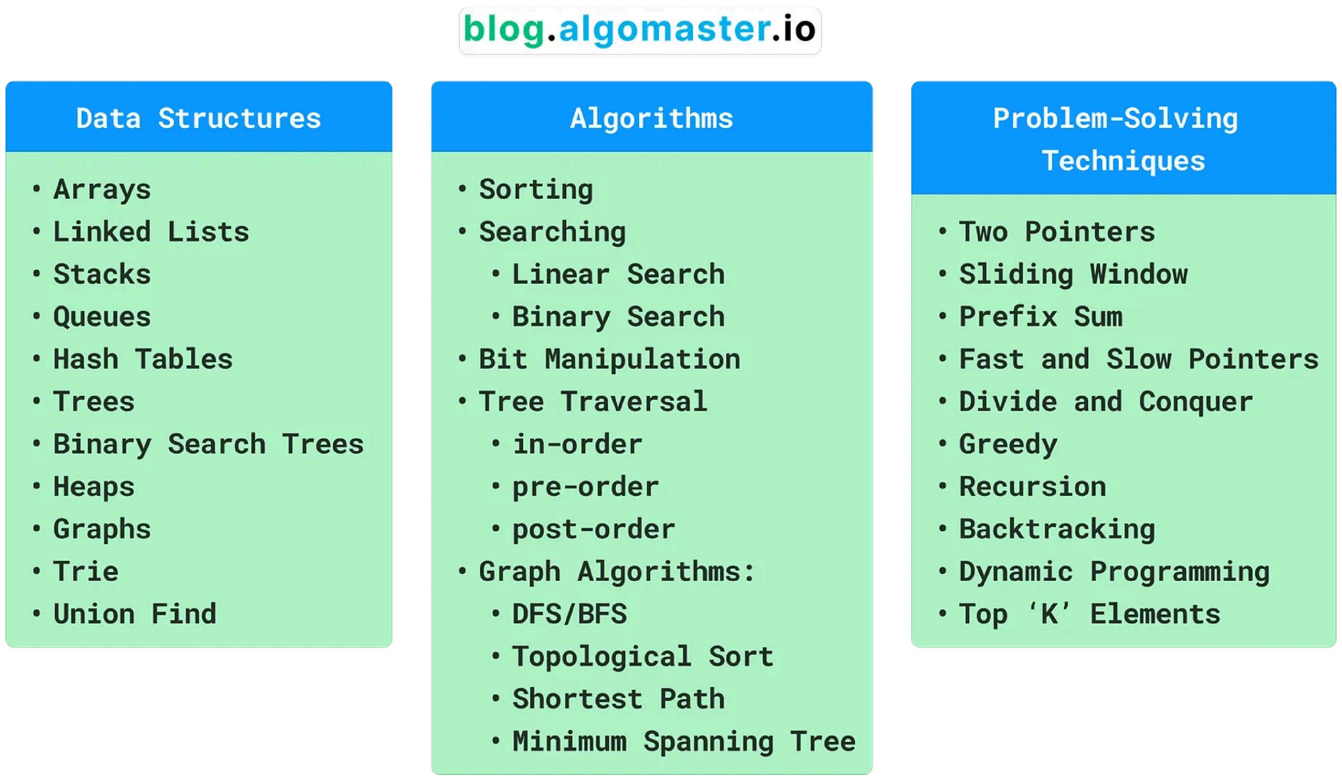
I'll discuss the key topics to focus on and the right order to learn them, how to start learning a new topic, resources you can use and how to scale your DSA knowledge.

I'll also share an effective revision strategy so that you don't forget the problems you've already solved.

1. Must Know Topics

First things first, let's talk about the key topics you need to focus on.

The three main pillars of DSA are **Data Structures**, **Algorithms**, **and Problem-Solving Techniques.**



There are other topics like Segment Trees, Fenwick Trees but they are rarely asked in coding interviews.

✅ In the beginning, focus on the most common topics.

2. Learn one topic at a time

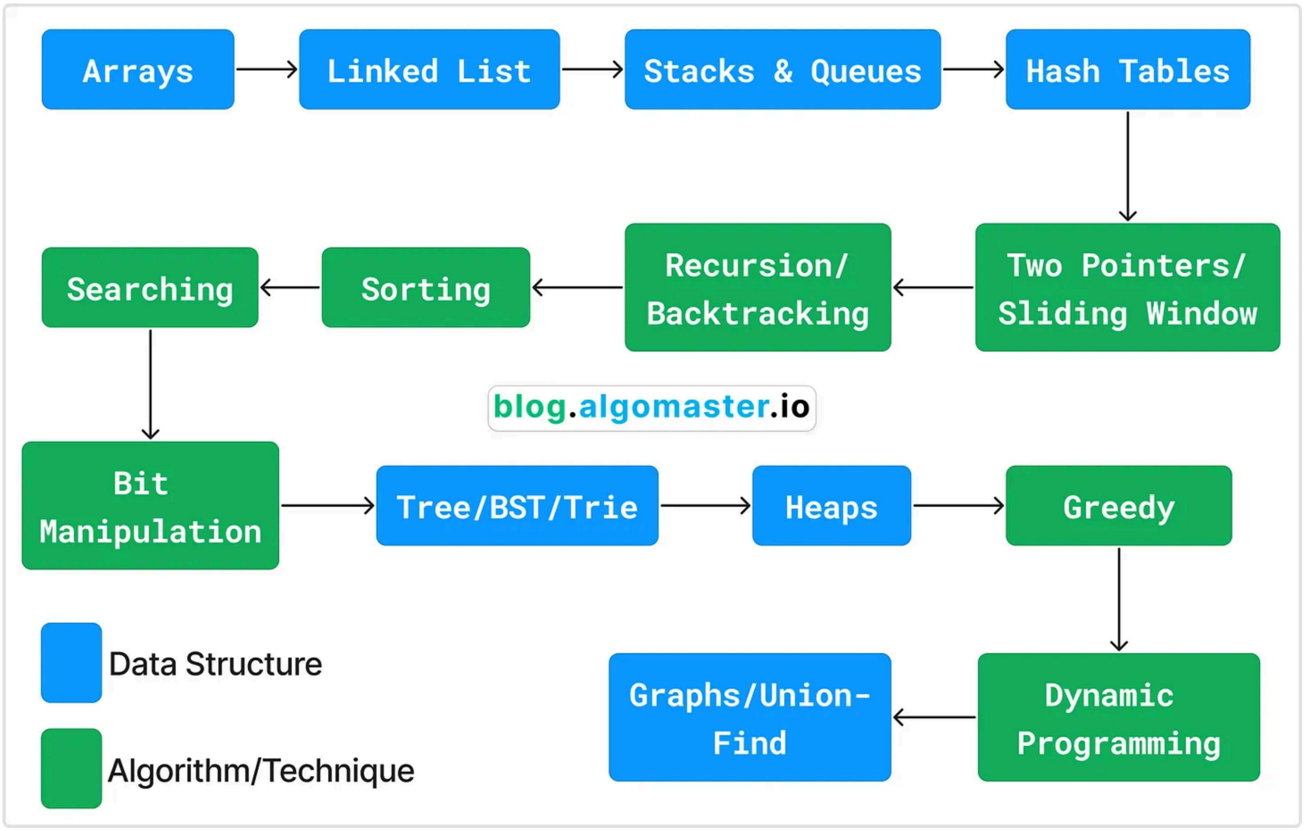
Trying to learn multiple topics simultaneously can be overwhelming and lead to confusion.

Focusing on one topic at a time makes the learning process more manageable and less stressful.

You should start with easy topics first and gradually move to difficult topics.

Begin with **linear data structures** like arrays, linked lists, stack, queues before moving on to more complex ones like trees, heaps or graphs.

Here is an order you can follow:



Now, lets talk about how to start learning a new topic.

3. How to start learning a new topic?

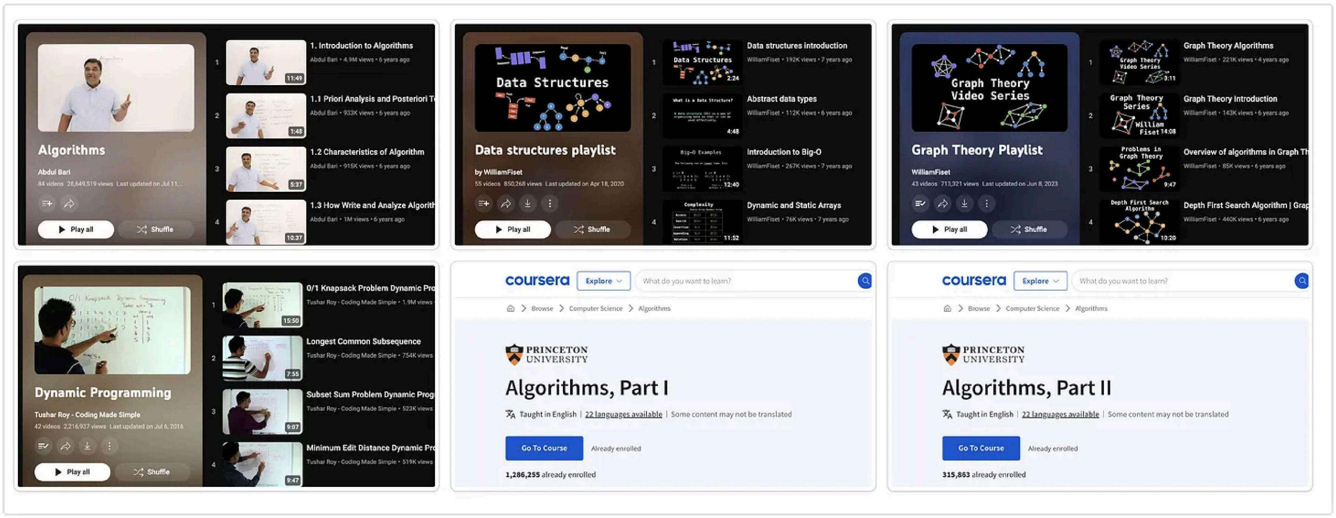
- **Start with Basics:** Start by learning what it is, how it's represented in code, different operations you can perform on it and their time/space complexities.
    - **Example:** For binary search trees, understand how it’s represented and learn common operations like inserting a node, deleting a node, and searching for a node.
- **Real-world applications:** Understand where and how it is used in real-world scenarios.
    - **Example:** Graphs are used in routing algorithms to find the shortest path between nodes in a network, such as in GPS navigation systems.
- **Use Pen and Paper:** Visualizing concepts on pen and paper helps in better understanding and retention. Draw diagrams, flowcharts, and write pseudocode to visualize how the data structure or algorithm works.
    - **Example:** For linked lists, draw nodes and pointers to see how nodes are connected and how operations like insertion and deletion are performed.
- **Implement it from Scratch:** Write code to implement the data structure or algorithm from scratch. Implementing it yourself reinforces your understanding and helps you grasp the underlying mechanics.
    - **Example:** Implement a stack using arrays or linked lists. Write functions for push, pop, and peek operations.
- **Learn the Inbuilt Library Functions:** Learn how to use standard libraries in your programming language that provide data structures and algorithms.
    - **Example:** In Python, learn how to use lists, sets, and dictionaries. In Java, get familiar with ArrayList, HashMap, and TreeSet.
- **Solve Simple Problems:** Just reading about a topic or watching tutorials is not enough. Practice simple problems related to the topic to build confidence and reinforce learning.
    - Solve 4-5 easy problems on LeetCode from the topic you are currently learning.

Subscribe to receive new articles every week.

4. Resources

There are many great resources available online.

Here are some that I personally found quite useful to build a solid foundation in DSA:


Algorithms
Abdul Bari
1. Introduction to Algorithms
1.1 Priori Analysis and Posteriori Testing
1.2 Characteristics of Algorithm
1.3 How Write and Analyze Algorithm
Data Structures
Data structures playlist
by WilliamFiset
Data structures introduction
Abstract data types
Introduction to Big-O
Dynamic and Static Arrays
Graph Theory Video Series
Graph Theory Playlist
WilliamFiset
Graph Theory Algorithms
Graph Theory Introduction
Overview of algorithms in Graph Theory
Depth First Search Algorithm | Graph Theory
Dynamic Programming
Tushar Roy - Coding Made Simple
0/1 Knapsack Problem Dynamic Programming
Longest Common Subsequence
Subset Sum Problem Dynamic Programming
Minimum Edit Distance Dynamic Programming
Play all
Shuffle
coursera
Explore
What do you want to learn?
PRINCETON UNIVERSITY
Algorithms, Part I
Go To Course
1,286,255 already enrolled
Algorithms, Part II
Go To Course
315,863 already enrolled

- [Abdul Bari's Algorithms Playlist]
- [William Fiset's Data Structure Playlist]
- [William Fiset's Graphs Playlist]
- [Tushar Roy's Dynamic Programming Playlist]
- [Coursera - Algorithms 1]
- [Coursera - Algorithms 2]

You can find more DSA resources in this [GitHub repository].

5. How to scale your DSA knowledge?

After you have learned the basics, you are ready to go deep.

Prioritize solving problems over theory

While theoretical understanding is important in the beginning, the real learning happens when you apply that knowledge to solve problems.

The more problems you solve, the more you'll reinforce your understanding of data structures and algorithms.

Challenge yourself

Getting good at DSA is similar to building muscles in the gym.

Lifting the same weights everyday won't make you stronger. You need to slowly lift heavier weights to build muscle.

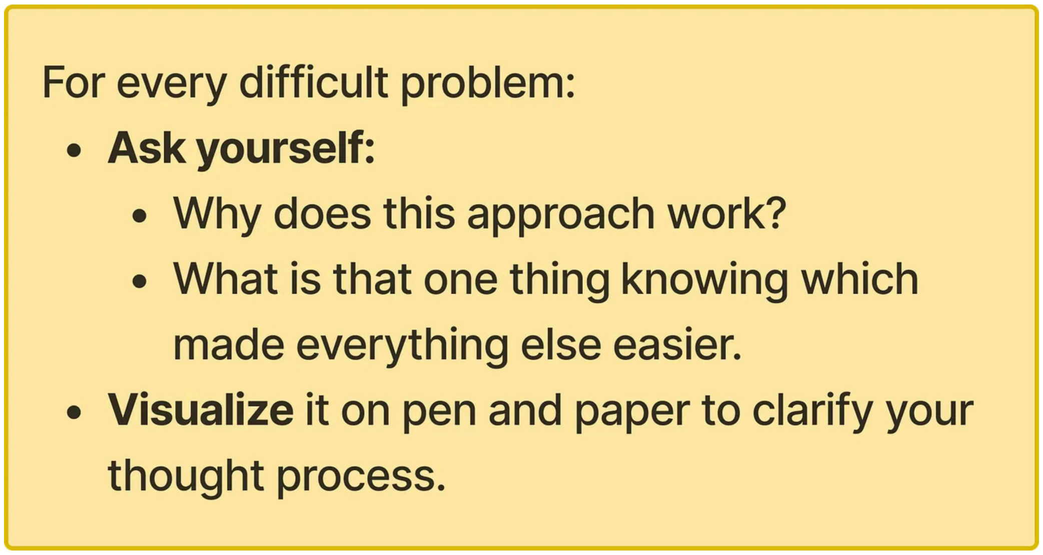
Likewise, to get better at DSA, you should slowly tackle harder problems.

> **Example:** If you can comfortably solve most easy problems, move to medium problems.

Understand, don't memorize

When learning DSA, focus on understanding the underlying concepts and principles rather than just memorizing code or solutions.

Memorization may help you solve specific problems, but it limits your ability to adapt and apply your knowledge to new scenarios.



For every difficult problem:

- **Ask yourself:**
  - Why does this approach work?
  - What is that one thing knowing which made everything else easier.
- **Visualize** it on pen and paper to clarify your thought process.

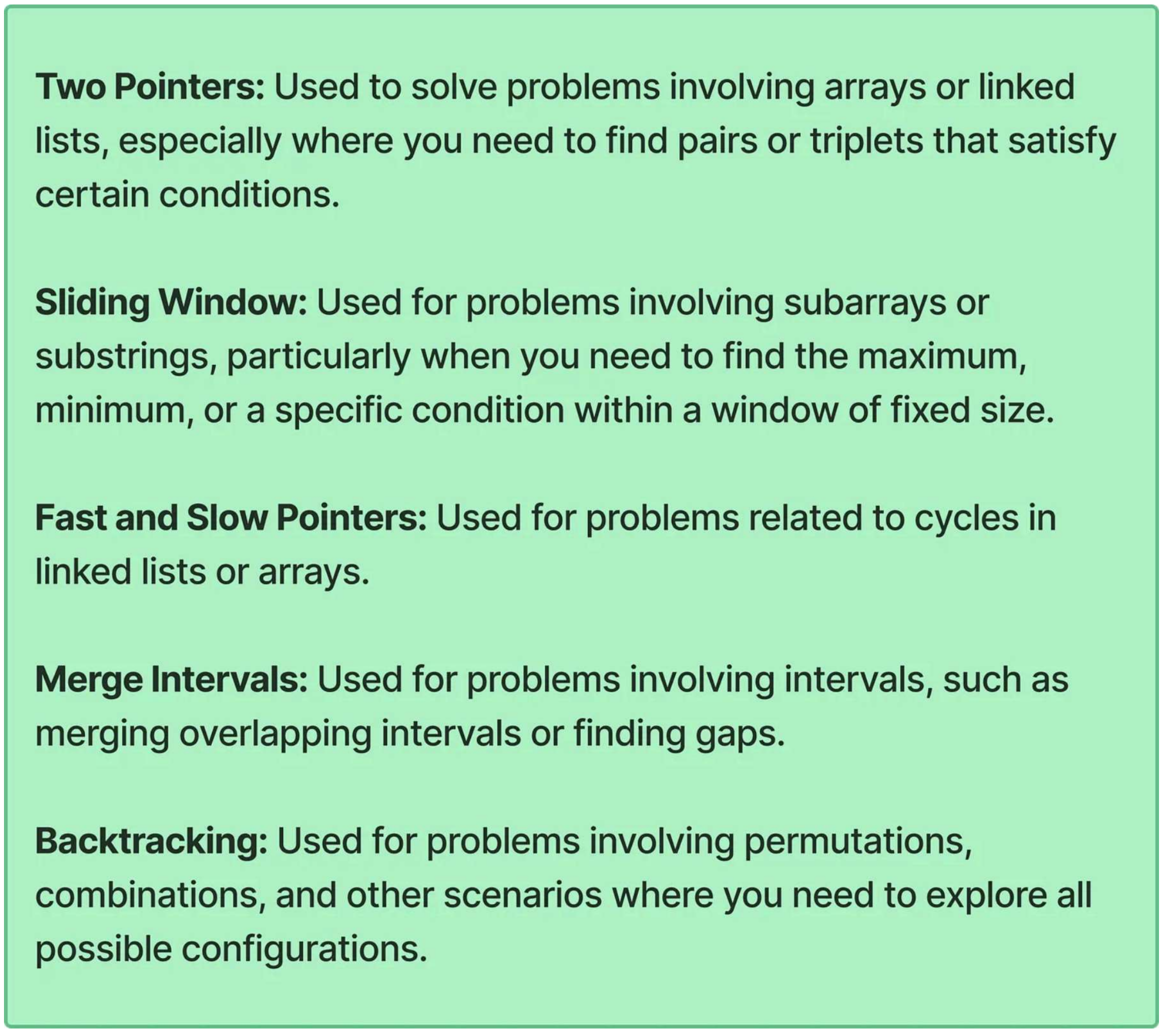
Think in Patterns

Many problems share underlying patterns. Recognizing these patterns can help you quickly identify the right approach to solve new problems, reducing the time spent on trial and error.

As you solve more problems, take note of recurring techniques and approaches. Group similar problems together and identify the common strategies used.

Understanding patterns enables you to adapt solutions to different but related problems.

Some of the common problem-solving patterns are:



Two Pointers: Used to solve problems involving arrays or linked lists, especially where you need to find pairs or triplets that satisfy certain conditions.

Sliding Window: Used for problems involving subarrays or substrings, particularly when you need to find the maximum, minimum, or a specific condition within a window of fixed size.

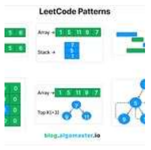
Fast and Slow Pointers: Used for problems related to cycles in linked lists or arrays.

Merge Intervals: Used for problems involving intervals, such as merging overlapping intervals or finding gaps.

Backtracking: Used for problems involving permutations, combinations, and other scenarios where you need to explore all possible configurations.

I wrote another article on **15 Most Common LeetCode Patterns.**

Check it out here:



LeetCode was HARD until I Learned these 15 Patterns

ASHISH PRATAP SINGH · 21 JUL

Read full story →

6. Retain what you Learn

Mastering DSA requires not only learning new concepts and solving problems but also retaining that knowledge over time.

Repetition is key

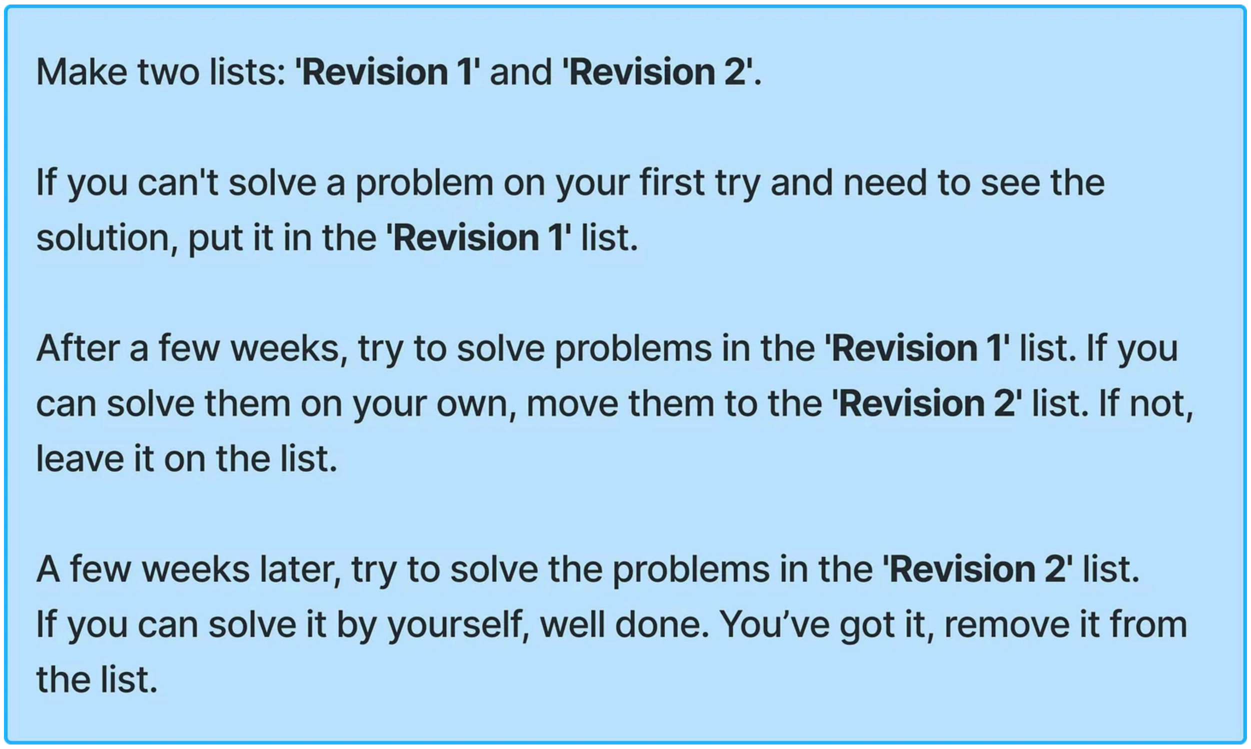
Repetition is key to transferring knowledge from **short-term** to **long-term memory**.

By revisiting concepts and problems regularly, you reinforce your understanding and make it easier to recall the information later.

Regularly revisit problems you found challenging and try to solve them again without looking at solutions.

Create Revision List

On LeetCode, you can make lists.



Make two lists: **'Revision 1'** and **'Revision 2'**.

If you can't solve a problem on your first try and need to see the solution, put it in the **'Revision 1'** list.

After a few weeks, try to solve problems in the **'Revision 1'** list. If you can solve them on your own, move them to the **'Revision 2'** list. If not, leave it on the list.

A few weeks later, try to solve the problems in the **'Revision 2'** list. If you can solve it by yourself, well done. You've got it, remove it from the list.

I found this to be an effective approach for the problems I couldn’t solve in the first attempt.

Bookmark anything important

Having quick access to high-quality resources saves time and helps you revisit important concepts and solutions.

Use browser bookmarks or tools like Google Drive, Notion to save links to useful articles, tutorials, and problem-solving guides. Tag and organize bookmarks by category for easy access.

You can find my most important resources in this GitHub repository.

7. Be Consistent

Learning DSA takes time.

Some topics might take weeks or months to master. Be patient with yourself.

And, it's normal to feel stuck or frustrated when solving a challenging problem or trying to understand a complex topic.

Keep going.

If a problem seems too hard, take a break, then try again.

If you need help, check hints or use the LeetCode discussion forum.

Thank you so much for reading.

If you found it valuable, hit a like ❤️ and consider subscribing for more such content every week.

If you have any questions or suggestions, leave a comment.

This post is public so feel free to share it.

Subscribe for free to receive new posts and support my work.

Checkout my Youtube channel for more in-depth content.

Follow me on LinkedIn, X and Medium to stay updated.

Checkout my GitHub repositories for free interview preparation resources.

I hope you have a lovely day!

See you soon,

Ashish

620 Likes · 47 Restacks

Previous Next

Discussion about this post

Comments Restacks

Write a comment...

Manoj kumar 16 Jun

Liked by Ashish Pratap Singh

I think It would be better in this order?

1. Array

2. String and string buffer if Java

3. Sorting

4. Searching

5. LinkedIn list

5. Stack and queue

6. Recursion

7.Two pointers and fast slow pointer

8. Bit manipulation

9. Tree

10.Graph

11. Divine and conquer

12. Backtracking

13. Dynamic problem

Then additional

It really good pieces captured to be master in DSA and problem solving.

Especially I love these highlights points

- One topic one time

- Retention like review 1 and revision 2 my favourite

- consistency

- pen and A4 blank page or white board

- Write code notepad or Google docs no any auto suggestions or highlighter

LIKE (12) REPLY SHARE

E

Ezekiel Kouassi 6 Jul

Liked by Ashish Pratap Singh

Thank you!

Now I Known how to master DSA. Continue sharing, you help a lot.

LIKE (3) REPLY SHARE

1 reply by Ashish Pratap Singh

19 more comments...