Rakesh Kumar R

1 Day DSA Prep Guide

✍️ A Note from Me

Hey! Before you dive into this guide, I just want to share a little something.

This isn't some expert-made, ultimate DSA roadmap — it's a simple plan I put together, genuinely thinking about what would help someone who's just getting started or has very little time to prep for placements.

The links I've shared throughout (whether they're videos, articles, or problems) are from various reputed sources and platforms. I've only added them as reference points, not as "must-follow" materials. You're totally free to learn from whichever source you're comfortable with. The idea is to give you direction, not restriction.

I made this guide not just for you, but also for my friends and myself — something we can come back to during revision, last-minute prep, or even on a random weekend grind.

Hope this helps you get started and keeps things simple.

Let's do this together.

Rakesh Kumar R   

Step-1

Learn a Language

⭐ Choose a Language

Before you dive into solving problems, pick one programming language you'll stick with throughout your DSA journey. No switching midway.

Understand the pros & cons and choose the right language that matches your background, goals, and target companies.

Do you already know a language?

🟢YES → Great! Use that language for DSA.
🔴NO → Pick a Language of your Choice.

Learn Python

- Beginner-friendly syntax
- Focus more on logic, less on syntax
- Slower execution, but rarely a problem in interviews
- Use if you're just starting out or want fast results

Python English Tutorial: **Click to Watch**

Python Tamil Tutorial: **Click to Watch**

Python Cheatsheet: **Click to Read**

Learn Java

- Most used in service/product companies
- Powerful OOP concepts
- Rich library for data structures
- Use if you're aiming for Amazon, Flipkart, etc.

Java English Tutorial: **Click to Watch**

Java Tamil Tutorial: **Click to Watch**

Java Cheatsheet: **Click to Read**

Learn C++

- Fastest execution
- STL (Standard Template Library) helps with DSA
- Used in coding competitions
- Use if you're aiming for Codeforces, Google, etc.

C++ English Tutorial: **Click to Watch**

C++ Tamil Tutorial: **Click to Watch**

C++ Cheatsheet: **Click to Read**

Learn Javascript

- Easy if you already know JS
- Works well with browser-based problems
- Not common in interviews, but usable
- Use if you're from a frontend/backend background

Javascript English Tutorial: **Click to Watch**

Javascript Tamil Tutorial: **Click to Watch**

Javascript Cheatsheet: **Click to Read**

⭐ Learn Basic Concepts

Before you solve problems, you must know the building blocks of your chosen language. This step will cover just enough basics to get you started with DSA — no deep theory, only what you'll use 90% of the time.

Learn these 6 Concepts

1. Variables: What are variables?
2. Datatypes: What types of data can you store (int, float, string, list, etc.)
3. Input/Output (I/O): How to take input from the user, and how to print output
4. Conditionals: Use of if, else if, else statements to control flow
5. Loops: Use of for and while loops to repeat code
6. Functions: How to define and call a function, use parameters and return values

Basic Concepts Tamil Tutorial: **Click to Watch**

Step-2

Logic Building



⭐ Solve Basic Problems

Before jumping into DSA structures like arrays and strings, solve a few foundational problems to sharpen your thinking and get comfortable with writing logic from scratch.

We'll solve 3 categories of problems. Each has 5 handpicked beginner-friendly Problems that build your logic skills.

- Math's Problems
- Recursion Problems
- Star Pattern Problems

Math's Problems

These Problems improve logic, arithmetic operations, and loop mastery.

1. Check Prime: Return true if a number is prime.

Click to Solve **Click to Learn**

2. Factorial: Calculate factorial of a number using loop.

Click to Solve **Click to Learn**

3. Count Digits: Count number of digits in an integer.

Click to Solve **Click to Learn**

4. Reverse a Number: Reverse a number (e.g. 123 → 321).

Click to Solve **Click to Learn**

5. GCD of Two Numbers: Use Euclidean algorithm to find GCD.

Click to Solve **Click to Learn**

Recursion Problems

Learn how to write functions that call themselves — a critical DSA concept.
Learn Recursion - **Watch**

1. Factorial using Recursion: Return factorial of n recursively.
Click to Solve **Click to Learn**

2. Fibonacci Number: Return nth Fibonacci number using recursion.
Click to Solve **Click to Learn**

3. Print 1 to N: Print numbers from 1 to N recursively.
Click to Solve **Click to Learn**

4. Power (a^b): Write a function to calculate a^b.
Click to Solve **Click to Learn**

5. Sum of Array Recursively: Return sum of array using recursion.
Click to Solve **Click to Learn**

☆ Star Pattern Problems

Improve loop logic, nested structure handling, and spatial thinking.

1. Right Half Pyramid **Click to Learn**
2. Left Half Pyramid **Click to Learn**
3. Full Pyramid **Click to Learn**
4. Inverted Full Pyramid **Click to Learn**
5. Hollow Rectangle **Click to Learn**

Watch this video to learn Star Pattern Problems Clearly.

Click to Watch

Step-3

What is DSA?



⭐ What is DSA?

DSA = Data Structures + Algorithms

Data Structures = Ways to store and organize data efficiently.

Algorithms = Step-by-step logic to solve a problem using those structures.

Together, they help in solving complex problems efficiently.

⭐ Why DSA Matters:

- Every top tech company asks DSA-based questions.
- It Makes your code faster and smarter.
- From search engines to games — all use DSA.
- Learning DSA is not just theoretical Knowledge — it's about Building problem solving mindset.

⭐ Types of Data Structures

There are two main types of data structures

Linear **Non-Linear**

Linear Data Structures: Elements are arranged one after another. They are Array, Linked List, Stack, Queue.

Non-Linear Data Structures: Elements are connected in hierarchical or graph structures.. They are Trees, Graphs, Heaps, Tries.

⭐ Algorithms

These are the Algorithms that are in DSA which are used to solve Problems efficiently.

1. Searching - Binary Search, Linear Search
2. Sorting - Bubble, Merge, Quick, Insertion
3. Recursion - Backtracking, Divide & Conquer
4. Greedy - Interval problems, coin change
5. Dynamic Programming - Tabulation and Memoization
6. Graph Algorithms - BFS, DFS, Dijkstra, Union-Find

Step-4

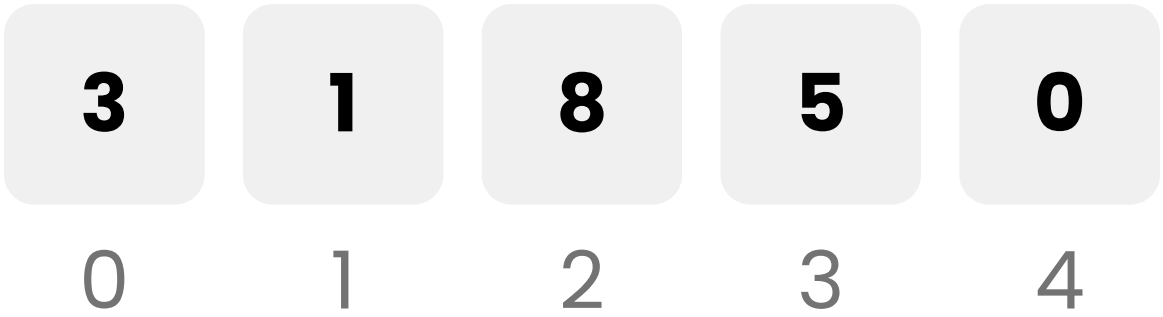
Arrays



⭐ What is an Array?

Imagine a row of boxes placed next to each other — each box can hold a number, a letter, or any piece of data.

That's what an array is — a fixed-size, ordered collection of elements, all of the same type, and each element is placed at a specific position called an index.



Indexes usually start from zero. So, in an array of 5 elements, the first item is at position 0, and the last is at position 4.

Once you understand how arrays work, start solving problems. Start with easy-to-medium problems that test your ability to write loops, use conditions, and think in steps.

Second Largest Element in an Array

Click to Solve **Click to Learn**

Reverse an Array In-Place

Click to Solve **Click to Learn**

Check if Array is Sorted

Click to Solve **Click to Learn**

Remove Duplicates from Sorted Array

Click to Solve **Click to Learn**

Move All Zeros to the End

Click to Solve **Click to Learn**

Two Sum

Click to Solve **Click to Learn**

Maximum Subarray Sum

Click to Solve **Click to Learn**

Step-5

Strings



⭐ What is a String?

A string is simply a sequence of characters. Think of it like an array of letters, digits, symbols, or even spaces.

For example, "hello" is a string of 5 characters.

You access each character using an index (like arrays).

Once you understand how Strings work, start solving problems.

Reverse a String

Click to Solve **Click to Learn**

Check if a String is a Palindrome

Click to Solve **Click to Learn**

Reverse words in a string

Click to Solve **Click to Learn**

Check if Strings Are Rotations of Each Other

Click to Solve **Click to Learn**

First non-repeating character of given string

Click to Solve **Click to Learn**

Roman to Integer

Click to Solve **Click to Learn**

Isomorphic Strings Check

Click to Solve **Click to Learn**

Check if two Strings are Anagrams of each other

Click to Solve **Click to Learn**

Conclusion



❌ This 1-Day DSA Plan won't make you a master of DSA.

But...

It will do something far more important if you're just starting out or low on time.

It acts as a Launchpad — a well-structured starting point for anyone who:

- Doesn't know where to begin
- Has minimal time before interviews or placements
- Feels overwhelmed by hundreds of DSA topics online

This guide gives clarity, not confusion. It focuses on real interview patterns and avoids random theory dumps.

⭐ What to do After Day 1?

If you're serious about improving:

- Move to Linked Lists, Hashing, Sliding Window, Binary Search, etc.
- Pick one DSA pattern each day and go deeper.
- Practice on platforms like LeetCode, CodeStudio, GFG, or HackerRank.

⭐ Final Thought

You don’t need 1000 problems to get placed.

You need clarity, consistency, and practice on the right problems.

the End