PROFESSIONAL DSA PROGRAM | INTERVIEW-FOCUSED CURRICULUM

Learning Roadmap

A visual path through the complete 8-module curriculum.

2 Months	8 Modules	DSA + Practice
Program Duration	Complete Curriculum	Applied Learning

01 **Foundations & Complexity Analysis** Foundation	02 **Arrays, Strings & Hashing** Core Data Structures
03 **Recursion, Searching & Sorting** Core Algorithms	04 **Linked Lists, Stacks & Queues** Linear Data Structures
05 **Trees, Heaps & Tries** Non-linear Data Structures	06 **Graphs** Non-linear Data Structures
07 **Greedy & Dynamic Programming** Advanced Techniques	08 **Interview Preparation & Mock Tests** Interview Prep

Program at a Glance

Key information about the Learn DSA program.

Program	Learn DSA (Data Structures & Algorithms) Program
Duration	2 Months (approx. 8 weeks), 8 modules
Learning Approach	Instructor-led sessions, daily problem solving, module assignments and mock interviews
Eligibility	Graduates and final-year students from any stream; basic programming knowledge (variables, loops, functions) is recommended
Technology Stack	C++ (STL), VS Code, Git and GitHub, Online Judges (LeetCode / GeeksforGeeks)
Program Outcome	Confident problem solver who can choose the right data structure and algorithm and perform well in coding rounds and technical interviews

Program Structure & Timeline

Module-wise duration, phase and primary tools, progressing from programming basics to advanced techniques and interview preparation.

No.	Module	Phase	Duration	Key Tools
01	Foundations & Complexity Analysis	Foundation	1 Week	C++ (STL), VS Code, Git, Online Judge
02	Arrays, Strings & Hashing	Core Data Structures	1 Week	Arrays, Strings, Hash Map, Hash Set
03	Recursion, Searching & Sorting	Core Algorithms	1 Week	Recursion, Binary Search, STL sort
04	Linked Lists, Stacks & Queues	Linear Data Structures	1 Week	Pointers, STL stack / queue / deque
05	Trees, Heaps & Tries	Non-linear Data Structures	1 Week	Binary Tree, BST, Priority Queue, Trie
06	Graphs	Non-linear Data Structures	1 Week	Adjacency List, BFS, DFS, Dijkstra, DSU
07	Greedy & Dynamic Programming	Advanced Techniques	1 Week	Greedy, Memoization, Tabulation
08	Interview Preparation & Mock Tests	Interview Prep	1 Week	Online Judge, Mock Interviews, GitHub
	Total Program Duration		**8 Weeks**	**Approx. 2 months**

MODULE 01

Foundations & Complexity Analysis

01

Programming refresher, how to measure the efficiency of code and a repeatable approach to solving problems.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Beginner**	**C++ (STL), VS Code, Git, Online Judge**	**A ready coding environment and the ability to analyze time and space complexity**

Topics Covered

Part 1: Programming Refresher

- Variables and Data Types
- Conditions and Loops
- Functions
- Pointers and References
- Input / Output Setup
- STL Basics (vector, pair, string)

Part 2: Complexity Analysis

- Why DSA Matters
- Time and Space Complexity
- Big-O, Omega, Theta
- Best, Average, Worst Case
- Analyzing Loops
- Constraints to Complexity

Part 3: Problem Solving Approach

- Understanding the Problem
- Dry Run and Edge Cases
- Pattern Recognition
- Debugging Techniques
- Basic Maths (GCD, Primes, Modulo)
- Using Online Judges

Setup Checklist

- Compiler and VS Code setup
- Git and GitHub account
- LeetCode / GeeksforGeeks profile

Practice Problems

- Pattern Printing
- Sieve of Eratosthenes
- GCD and Prime Checks
- Complexity Estimation Exercises

WHAT THIS MODULE SUPPORTS

ANALYZE	PREPARE	APPROACH
Estimate the cost of any solution before coding it.	Set up tools and a daily practice routine.	Break a problem into steps and test edge cases.

MODULE 02

Arrays, Strings & Hashing

02

The most frequently tested data structures, with the patterns that turn brute force into efficient solutions.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Beginner to Intermediate	Arrays, Strings, Hash Map, Hash Set	Solve array and string problems with efficient patterns

Topics Covered

Part 1: Arrays

- Traversal and Updates
- Prefix Sum
- Rotation and Reversal
- 2D Arrays and Matrices
- Kadane's Algorithm
- Sorting-based Solutions

Part 2: Core Techniques

- Two Pointers
- Frequency Counting
- Sliding Window (Fixed)
- Subarray Problems
- Sliding Window (Variable)
- Bit Manipulation Basics

Part 3: Strings

- String Manipulation
- Character Frequency
- Palindromes
- Pattern Matching Basics
- Anagrams
- Substring Problems

Part 4: Hashing

- Hash Maps and Hash Sets
- Subarray Sum with Hashing
- Collision Handling
- Grouping and Counting
- Two Sum Pattern
- Duplicates and Uniqueness

Practice Problems

- Two Sum
- Longest Substring Without Repeating Characters
- Maximum Subarray
- Group Anagrams

WHAT THIS MODULE SUPPORTS

OPTIMIZE	LOOKUP	RECOGNIZE
Replace nested loops with linear-time patterns.	Use hashing for constant-time search and counting.	Spot window and pointer problems quickly.

MODULE 03

Recursion, Searching & Sorting

03

Think recursively, search in logarithmic time and understand how classic sorting algorithms work.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Intermediate**	**Recursion, Binary Search, STL sort**	**Apply recursion, binary search and sorting to problems**

Topics Covered

Part 1: Recursion & Backtracking

- Recursion Basics
- Call Stack and Base Case
- Recursion Tree
- Subsets and Permutations
- Backtracking Basics
- N-Queens

Part 2: Searching

- Linear Search
- Binary Search
- First and Last Occurrence
- Search in Rotated Array
- Binary Search on Answer
- Search in 2D Matrix

Part 3: Sorting

- Bubble, Selection, Insertion
- Merge Sort
- Quick Sort
- Counting Sort
- Custom Comparators
- Stability of Sorting

Practice Problems

- Generate Subsets
- N-Queens
- Search in Rotated Sorted Array
- Merge Sort Implementation

WHAT THIS MODULE SUPPORTS

RECURSE	SEARCH	SORT
Break problems into smaller identical subproblems.	Cut search space in half at every step.	Choose the right sorting method for the data.

MODULE 04

Linked Lists, Stacks & Queues

04

Dynamic linear structures and the stack and queue patterns that appear again and again in interviews.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Intermediate**	**Pointers, STL stack / queue / deque**	**Implement and apply linear data structures confidently**

Topics Covered

Part 1: Linked List

- Singly, Doubly and Circular Lists
- Insert, Delete, Traverse
- Reverse a Linked List
- Fast and Slow Pointers
- Cycle Detection
- Merge Two Sorted Lists

Part 2: Stack

- Stack Operations
- Balanced Parentheses
- Next Greater Element
- Monotonic Stack
- Infix to Postfix
- Expression Evaluation

Part 3: Queue

- Queue and Circular Queue
- Deque
- Queue using Stacks
- Monotonic Deque
- Sliding Window Maximum
- Simulation Problems

Practice Problems

- Reverse Linked List
- Valid Parentheses
- Min Stack
- Sliding Window Maximum

WHAT THIS MODULE SUPPORTS

LINK	STACK	QUEUE
Manipulate nodes and pointers without errors.	Handle nested and order-dependent problems.	Process data in order with queues and deques.

MODULE 05

Trees, Heaps & Tries

05

Hierarchical data structures, from binary trees and BSTs to heaps and prefix trees.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Intermediate	Binary Tree, BST, Priority Queue, Trie	Traverse, search and optimize with tree-based structures

Topics Covered

Part 1: Binary Trees

- Tree Terminology
- DFS Traversals (In, Pre, Post)
- BFS / Level Order
- Height and Diameter
- Lowest Common Ancestor
- Left, Right and Top Views

Part 2: Binary Search Trees

- Search and Insert
- Delete a Node
- Validate a BST
- Inorder Successor
- Kth Smallest Element
- Balanced Trees Overview

Part 3: Heaps

- Min Heap and Max Heap
- Priority Queue
- Heapify and Heap Sort
- Top-K Problems
- Merge K Sorted Lists
- Median of a Stream

Part 4: Tries

- Trie Insert and Search
- Prefix Matching
- Autocomplete Use Case

Practice Problems

- Level Order Traversal
- Kth Largest Element
- Validate BST
- Implement Trie

WHAT THIS MODULE SUPPORTS

TRAVERSE	ORDER	MATCH
Visit every node with DFS and BFS.	Keep sorted data with BSTs and heaps.	Search prefixes quickly with tries.

MODULE 06

Graphs

06

Model real-world relationships as graphs and solve connectivity, ordering and shortest-path problems.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Intermediate to Advanced	Adjacency List, BFS, DFS, Dijkstra, DSU	Model problems as graphs and solve them with standard algorithms

Topics Covered

Part 1: Graph Fundamentals

- Graph Terminology
- Directed and Undirected Graphs
- Weighted Graphs
- Adjacency Matrix
- Adjacency List
- Graph Input Formats

Part 2: Traversals & Applications

- Breadth-First Search
- Depth-First Search
- Connected Components
- Cycle Detection
- Grid and Matrix Problems
- Bipartite Check

Part 3: Core Graph Algorithms

- Topological Sort
- Dijkstra's Shortest Path
- Union-Find (DSU)
- Minimum Spanning Tree (Kruskal's)
- Bellman-Ford (Introduction)

Practice Problems

- Number of Islands
- Course Schedule
- Shortest Path in a Weighted Graph
- Number of Provinces

WHAT THIS MODULE SUPPORTS

MODEL	EXPLORE	OPTIMIZE
Turn grids, maps and dependencies into graphs.	Choose between BFS and DFS for each problem.	Find shortest paths and minimum spanning trees.

MODULE 07

Greedy & Dynamic Programming

07

Optimization techniques: make locally best choices or reuse stored subproblem results to avoid repeated work.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Intermediate to Advanced	Greedy, Memoization, Tabulation	Recognize and solve greedy and dynamic programming problems

Topics Covered

Part 1: Greedy Algorithms

- Greedy Strategy
- Activity Selection
- Merge Intervals
- Fractional Knapsack
- Jump Game
- Proving a Greedy Choice

Part 2: DP Fundamentals

- Overlapping Subproblems
- Memoization (Top-Down)
- Tabulation (Bottom-Up)
- State and Transition
- Space Optimization

Part 3: DP Patterns

- 1D DP (Climbing Stairs, House Robber)
- 0/1 Knapsack and Subset Sum
- Coin Change
- Longest Common Subsequence
- Longest Increasing Subsequence
- Grid DP

Practice Problems

- Climbing Stairs
- Coin Change
- Longest Common Subsequence
- Longest Increasing Subsequence

WHAT THIS MODULE SUPPORTS

DECIDE	REMEMBER	DEFINE
Pick the best choice at each step when valid.	Store results so no subproblem is solved twice.	Frame a problem as states and transitions.

MODULE 08

Interview Preparation & Mock Tests

08

Consolidate every module through revision, timed rounds and mock interviews.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Applied Practice	Online Judge, Mock Interviews, GitHub	Interview-ready problem solving with a documented practice portfolio

Topics Covered

Part 1: Revision & Patterns

- Pattern Cheat Sheet
- Topic-wise Revision
- Mixed Problem Sets
- Common Mistakes and Fixes

Part 2: Interview Skills

- Explaining Your Approach
- Stating Complexity Clearly
- Handling Hints and Follow-ups
- Thinking Aloud
- Writing Clean Code

Part 3: Mock Rounds

- Timed Coding Rounds
- Mock Technical Interviews
- Company-style Problem Sets
- Final Assessment

Practice Problems

- Full-length Timed Mock Test
- Two Mock Interviews
- Personal Problem-Solving Repository on GitHub

WHAT THIS MODULE SUPPORTS

REVISE	COMMUNICATE	DOCUMENT
Revisit every pattern with mixed problem sets.	Explain solutions clearly under time pressure.	Show your progress in a public repository.

This curriculum is suitable for a 2-month intensive training program, progressing from programming basics and complexity analysis to advanced techniques and interview preparation.