
# DSA TRAINING

## BEST DATA STRUCTURES & ALGORITHMS TRAINING COURSE

DSA (Data Structures and Algorithms) is the foundation of efficient problem-solving and software development, covering how data is organized, stored and processed to build fast, scalable and optimized programs. This training is designed for students and beginners to understand DSA concepts and apply them through practical, real-world coding exercises.

## KEY FEATURES OF DSA COURSE

- Learn the fundamentals of Data Structures and Algorithms and how they power efficient, scalable software.
- Understand time and space complexity analysis using Big-O, Big-Omega and Big-Theta notation.
- Master linear data structures including Arrays, Strings, Linked Lists, Stacks and Queues.
- Understand non-linear data structures including Trees, Binary Search Trees, Heaps and Graphs.
- Learn core algorithm techniques including Recursion, Backtracking, Divide & Conquer, Greedy and Dynamic Programming.
- Practice Searching and Sorting algorithms and understand where and why each one is used.
- Work with commonly used tools and platforms including VS Code, GitHub, LeetCode, GeeksforGeeks and online visualizers.
- Learn problem-solving patterns and how to approach coding interview questions systematically.
- Understand hashing, collision handling and real-world use cases of hash-based data structures.
- Complete a practical DSA project involving problem-solving across multiple data structures and algorithms.

## WHY DSA TRAINING?

- Understand how efficient data structures and algorithms power real-world software and applications.
- Build practical skills that can be applied to coding interviews, competitive programming and software development roles.
- Learn theoretical concepts together with hands-on coding exercises and real-world problem-solving.
- Gain experience in complexity analysis, algorithm design, debugging and code optimization.
- Develop strong problem-solving and logical thinking skills through consistent practice.
- Stay prepared for technical interviews with structured, topic-wise practice and mock problem sessions.

## Course Syllabus

### Introduction to DSA

- Meaning and importance of Data Structures and Algorithms
- Linear vs Non-Linear Data Structures
- Time Complexity and Space Complexity
- Big-O, Big-Omega and Big-Theta Notation
- Best, Average and Worst Case Analysis
- Introduction to Problem-Solving Approach
- Setting up the Coding Environment

### Arrays & Strings

- Introduction to Arrays
- 1D and 2D Arrays
- Array Operations -- Insertion, Deletion, Traversal
- Two Pointer Technique
- Sliding Window Technique
- Prefix Sum and Difference Array
- Introduction to Strings
- String Manipulation and Pattern Problems
- Practical Array and String Problems

### Searching & Sorting

- Linear Search

- Binary Search and its Variations
- Bubble Sort, Selection Sort and Insertion Sort
- Merge Sort
- Quick Sort
- Counting Sort and Basic Non-comparison Sorts
- Time Complexity Comparison of Sorting Algorithms
- Practical Searching and Sorting Problems

### Linked List

- Introduction to Linked List
- Singly Linked List
- Doubly Linked List
- Circular Linked List
- Insertion, Deletion and Traversal Operations
- Reversing a Linked List
- Cycle Detection -- Floyd's Algorithm
- Practical Linked List Problems

### Stacks & Queues

- Introduction to Stack -- LIFO Concept
- Stack Operations and Applications
- Introduction to Queue -- FIFO Concept
- Circular Queue and Deque
- Priority Queue -- Introduction
- Monotonic Stack and Queue -- Basic Understanding
- Practical Stack and Queue Problems

### Recursion & Backtracking

- Introduction to Recursion
- Recursive Tree and Call Stack Understanding
- Base Case and Recursive Case Design
- Introduction to Backtracking
- Permutations and Combinations
- N-Queens and Sudoku-style Problems -- Introduction
- Practical Recursion and Backtracking Problems

### Trees

- Introduction to Trees and Terminology
- Binary Trees and Binary Tree Traversals
- Binary Search Tree (BST) -- Insertion, Deletion, Search
- Balanced Trees -- Basic Understanding (AVL Introduction)
- Height, Depth and Diameter of a Tree
- Heaps -- Min-Heap and Max-Heap
- Trie -- Introduction
- Practical Tree Problems

### Graphs

- Introduction to Graphs and Terminology
- Graph Representation -- Adjacency List and Matrix
- Breadth-First Search (BFS)

- Depth-First Search (DFS)
- Topological Sorting -- Introduction
- Shortest Path Algorithms -- Dijkstra Introduction
- Union-Find / Disjoint Set -- Basic Understanding
- Practical Graph Problems

### Hashing

- Introduction to Hashing
- Hash Functions and Hash Tables
- Collision Handling -- Chaining and Open Addressing
- HashMap and HashSet -- Practical Usage
- Real-world Applications of Hashing
- Practical Hashing Problems

### Greedy & Dynamic Programming

- Introduction to Greedy Algorithms
- Activity Selection and Interval Problems
- Introduction to Dynamic Programming
- Memoization vs Tabulation
- 1D and 2D Dynamic Programming Problems
- Knapsack Problem -- Introduction
- Longest Common Subsequence -- Introduction
- Practical Greedy and DP Problems

### Interview Preparation

- Common Coding Interview Patterns
- Approach to Solving a New Problem
- Writing Clean and Optimized Code
- Explaining Time and Space Complexity in Interviews
- Mock Problem-Solving Sessions
- Common Mistakes and How to Avoid Them
- Platform Practice -- LeetCode and GeeksforGeeks

### Practical Project

- Select a set of real-world problem statements
- Identify the right data structure for each problem
- Design and implement the algorithm with optimal complexity
- Test the solution with multiple edge cases
- Analyze and document time and space complexity
- Optimize the solution for better performance
- Practice explaining the approach and logic clearly
- Solve topic-wise problem sets across all modules
- Participate in a mock coding interview / problem-solving session
- Prepare and present a complete solved problem portfolio

## OUR TRAINING BENEFITS

- Gain knowledge from experienced professionals in the field.
- Learn theoretical concepts and gain practical coding experience at the same time.
- Training that provides real-world, hands-on experience to understand problem-solving approaches.

- Complete practical DSA problems and build an interview-ready problem portfolio.
- Receive a certificate upon successful completion of training.
- Get exposure to current coding platforms, tools and interview practices.
- Learning tools, updated curriculum and course materials will be provided.
- Opportunity to interact with trainers for guidance during the training.

**Suggested Training Duration: 30 Hours**

Theory & Concept Learning -- 18 Hours | Practical Problem Solving -- 10 Hours | Final Project & Mock Interview -- 2 Hours

**Litz Tech India Pvt Ltd**

Industry-oriented training in DSA, programming technologies and practical skill development.