DSA, Competitive Programming and MERN Full-Stack Development Bootcamp

Duration: 5 days (40 hours)

Day 1: Programming Logic, Problem Solving and Algorithm Foundations

Theory: 1.5 Hours
Labs/Practical: 6.5 Hours

Module 1.1: Programming and Problem-Solving Foundations

Concepts

- Understanding a programming problem
- Identifying input, output and constraints
- Breaking a problem into smaller steps
- Writing algorithms and pseudocode
- Dry-running and tracing programs
- Handling edge cases
- Debugging logical and runtime errors
- Introduction to online coding platforms

Practical Activities

- Configure the programming environment
- Execute programs using a selected language
- Convert problem statements into pseudocode
- Trace program execution manually
- Identify and correct logical errors
- Solve input/output-based coding exercises

Module 1.2: Time and Space Complexity

Concepts

- Need for algorithm analysis
- Time complexity
- Space complexity
- Big-O notation
- Best, average and worst cases
- Comparison of iterative and recursive solutions

Practical Activities

- Analyse the complexity of loops
- Compare linear and logarithmic solutions

- Measure execution time for different approaches
- Optimise inefficient code
- Identify hidden nested-loop complexity

Module 1.3: Arrays and Strings

Concepts

- One-dimensional and multidimensional arrays
- Traversal, insertion and deletion
- String representation and manipulation
- Character-frequency techniques
- In-place operations
- Common array and string patterns

Practical Labs

- Find minimum and maximum values
- Find the second-largest element
- Reverse an array
- Rotate an array
- Remove duplicate elements
- Find missing numbers
- Check palindrome strings
- Check anagrams
- Count character frequency
- Reverse words in a sentence
- Find the longest common prefix
- Find the maximum subarray sum

Module 1.4: Mathematical and Logical Problem Solving

Concepts

- Number manipulation
- Divisibility rules
- Prime numbers
- Greatest common divisor and least common multiple
- Modular arithmetic fundamentals
- Pattern-based problem solving

Practical Labs

- Prime-number validation
- Factorial and Fibonacci series
- Armstrong-number validation
- Number and digit reversal
- Greatest common divisor calculation

- Least common multiple calculation
- Pattern-printing exercises
- Count digits and calculate digit sums

Day 2: Core Data Structures and Algorithmic Techniques

Theory: 1.5 Hours
Labs/Practical: 6.5 Hours

Module 2.1: Searching and Sorting Algorithms

Concepts

- Linear search
- Binary search
- Bubble sort
- Selection sort
- Insertion sort
- Merge sort overview
- Quick sort overview
- Stability and in-place sorting
- Selecting an appropriate algorithm

Practical Labs

- Implement linear and binary search
- Find first and last occurrence of an element
- Search in a rotated sorted array
- Implement bubble, selection and insertion sort
- Compare sorting-algorithm performance
- Sort custom objects using multiple fields

Module 2.2: Linked Lists

Concepts

- Array versus linked list
- Singly linked list
- Doubly linked list overview
- Node creation and traversal
- Insertion and deletion
- Fast and slow pointer technique

Practical Labs

- Create a singly linked list
- Insert nodes at different positions
- Delete a node

- Search for a value
- Reverse a linked list
- Find the middle node
- Detect a cycle
- Merge two sorted linked lists

Module 2.3: Stacks and Queues

Concepts

- Stack operations
- Queue operations
- Circular queue overview
- Deque overview
- Stack and queue use cases
- Implementation using arrays and linked lists

Practical Labs

- Implement a stack
- Implement a queue
- Validate balanced parentheses
- Reverse a string using a stack
- Evaluate postfix expressions
- Design a minimum stack
- Implement a queue using stacks
- Generate binary numbers using a queue

Module 2.4: Hashing and Problem-Solving Patterns

Concepts

- Hash tables
- Hash maps and hash sets
- Frequency counting
- Collision overview
- Two-pointer technique
- Sliding-window technique
- Prefix-sum technique

Practical Labs

- Solve the two-sum problem
- Find duplicate elements
- Find the first non-repeating character
- Group anagrams
- Find the longest substring without repeated characters
- Find pairs with a target sum

- Calculate range sums
- Find the maximum sum of a fixed-size subarray
- Solve subarray-sum problems

Day 3: Advanced DSA and Competitive Programming

Theory: 1.5 Hours
Labs/Practical: 6.5 Hours

Module 3.1: Recursion and Backtracking

Concepts

- Recursive problem decomposition
- Base and recursive cases
- Call-stack behaviour
- Recursion tree
- Introduction to backtracking
- Decision-space exploration
- Pruning unnecessary solutions

Practical Labs

- Factorial using recursion
- Fibonacci using recursion
- Generate all subsequences
- Generate permutations
- Solve subset-sum problems
- Solve maze-path problems
- Solve the N-Queens problem
- Generate balanced parentheses

Module 3.2: Trees, Binary Search Trees and Heaps

Concepts

- Tree terminology
- Binary trees
- Binary search trees
- Tree traversals
- Height and depth
- Heaps and priority queues
- Tree-based problem-solving patterns

Practical Labs

- Create a binary tree
- Implement preorder, inorder and postorder traversal

- Implement level-order traversal
- Find tree height
- Search and insert in a binary search tree
- Validate a binary search tree
- Find the lowest common ancestor
- Find the Kth largest or smallest element using a heap

Module 3.3: Graph Fundamentals

Concepts

- Graph terminology
- Directed and undirected graphs
- Weighted and unweighted graphs
- Adjacency matrix and adjacency list
- Breadth-first search
- Depth-first search
- Connected components
- Shortest-path overview

Practical Labs

- Represent a graph using an adjacency list
- Implement breadth-first search
- Implement depth-first search
- Count connected components
- Detect cycles
- Solve grid and island-counting problems
- Find the shortest path in an unweighted graph

Module 3.4: Greedy Algorithms and Dynamic Programming

Concepts

- Greedy-choice approach
- Recognising greedy problems
- Overlapping subproblems
- Optimal substructure
- Memoization
- Tabulation
- Comparing recursion and dynamic programming

Practical Labs

- Activity-selection problem
- Minimum-coin problem
- Fibonacci using memoization and tabulation
- Climbing-stairs problem

- House-robber problem
- Knapsack introduction
- Longest common subsequence introduction

Module 3.5: Competitive Programming Techniques

Concepts

- Reading constraints before coding
- Selecting suitable data structures
- Fast input and output
- Handling multiple test cases
- Avoiding time-limit exceeded errors
- Avoiding integer overflow
- Testing edge cases
- Writing concise and reusable code

Practical Activities

- Easy-to-medium timed coding contest
- Debugging incorrect submissions
- Optimising time-limit exceeded solutions
- Peer review of alternative approaches
- Post-contest solution discussion

Day 4: Front-End Web Development, JavaScript and ReactJS

Theory: 1.5 Hours
Labs/Practical: 6.5 Hours

Module 4.1: HTML and Web Development Foundations

Concepts

- Client-server architecture
- Structure of a web application
- Semantic HTML
- Forms and input controls
- Tables, lists and media elements
- Web accessibility fundamentals
- Browser developer tools

Practical Labs

- Create a multi-section web page
- Build a student-registration form
- Add appropriate input validation
- Create accessible navigation

- Inspect and debug HTML using developer tools

Module 4.2: CSS and Responsive Web Design

Concepts

- CSS selectors
- Box model
- Display and positioning
- Flexbox
- CSS Grid
- Media queries
- Responsive design
- Transitions and animations
- Reusable styling practices

Practical Labs

- Style a registration form
- Build a responsive navigation bar
- Create card-based layouts
- Design pages using Flexbox
- Design dashboards using CSS Grid
- Adapt layouts for mobile, tablet and desktop
- Add simple animations and transitions

Module 4.3: JavaScript and DOM Programming

Concepts

- Variables and data types
- Functions and arrow functions
- Arrays and objects
- Array methods
- DOM selection and manipulation
- Event handling
- Form validation
- Local storage
- Error handling
- Promises and asynchronous programming
- Fetch API
- JSON processing

Practical Labs

- Validate registration forms
- Dynamically add and remove records
- Search, sort and filter displayed data

- Store application data in local storage
- Consume data from a public API
- Display loading and error states
- Convert API responses into user-interface elements

Module 4.4: ReactJS Fundamentals

Concepts

- React application structure
- Component-based architecture
- Functional components
- JSX
- Props and state
- Event handling
- Conditional rendering
- Lists and keys
- Controlled forms
- React hooks
- API integration
- Client-side routing overview

Practical Labs

- Set up a React application
- Create reusable components
- Build navigation and page layouts
- Create controlled forms
- Implement search and filtering
- Manage state using hooks
- Fetch and display API data
- Add routing between application pages

Day 5: Node.js, Express, MongoDB and MERN Application Development

Module 5.1: Node.js and Express Fundamentals

Concepts

- Node.js runtime
- NPM and package management
- Modules
- Asynchronous programming
- Express application structure
- Routes and controllers
- Middleware
- Request and response processing

- Environment variables

Practical Labs

- Create a Node.js application
- Configure an Express server
- Create application routes
- Implement custom middleware
- Process query and route parameters
- Handle JSON request bodies
- Configure environment variables
- Implement centralised error handling

Module 5.2: REST API Development

Concepts

- REST architecture
- Resources and endpoints
- HTTP methods
- HTTP status codes
- CRUD operations
- Request validation
- Error responses
- Pagination and filtering
- API testing
- Authentication overview

Practical Labs

- Create GET, POST, PUT and DELETE APIs
- Add request validation
- Add searching and filtering
- Implement pagination
- Return appropriate status codes
- Handle application errors
- Test APIs using Postman
- Create a reusable Postman collection

Module 5.3: MongoDB and Mongoose

Concepts

- Relational versus document databases
- Documents and collections
- MongoDB CRUD operations
- Schema design
- Mongoose models

- Validation
- Referencing and embedding
- Querying and sorting
- MongoDB aggregation overview

Practical Labs

- Configure a MongoDB database
- Connect Node.js with MongoDB
- Create Mongoose schemas and models
- Insert and retrieve records
- Update and delete records
- Add schema validation
- Implement filtering and sorting
- Create basic aggregation queries

Module 5.4: Full-Stack MERN Integration

Practical Activities

- Connect the React frontend with Express APIs
- Connect Express with MongoDB
- Implement create, read, update and delete workflows
- Add client-side and server-side validation
- Handle loading, success and error states
- Configure cross-origin resource sharing
- Test frontend and backend integration
- Debug browser, server and database errors

Module 5.5: Integrated Capstone Project

Project: Student Skill and Opportunity Portal

User Management

- Student registration
- Profile creation
- Skills and interests
- Login workflow overview

Opportunity Management

- Add opportunities
- View available opportunities
- Search and filter records
- Save or apply for opportunities

Administration

- Add, edit and remove records
- Review registered students
- Update application status
- Generate basic reports