Complete DSA Mastery Schedule

16-Week Comprehensive Program for Data Structures & Algorithms

---

PROGRAM OVERVIEW

Goal: Master Data Structures & Algorithms with deep logical thinking and quick problem-solving skills

Duration: 16 Weeks (4 Months)

Daily Commitment: 2-4 hours

Target: From beginner to advanced level with strong interview preparation

---

PHASE 1: FOUNDATIONS (Weeks 1-4)

Week 1: Analysis & Mathematics

Daily Time: 2-3 hours

Learning Objectives:

- Master time and space complexity analysis
- Understand Big O, Omega, Theta notations
- Build strong mathematical foundation for algorithms

Daily Schedule:

- **Monday-Tuesday**:
  - Analysis of Algorithms (Big O, Omega, Theta)
  - Asymptotic analysis and order of growth
  - Analysis of loops and recursion
- **Wednesday-Thursday**:
  - Recursion basics and recursion tree method
  - Solving recurrences using recursion tree
  - Space complexity analysis
- **Friday**:
  - Mathematics problems (Count digits, palindrome, factorial)
  - GCD, LCM, prime numbers, Sieve of Eratosthenes

- **Saturday**:
  - Practice all mathematics problems from course
  - Implement algorithms from scratch
- **Sunday**:
  - Review week's concepts
  - Solve 3-5 LeetCode Easy recursion problems

Key Topics: Time Complexity, Space Complexity, Mathematical Algorithms

---

Week 2: Deep Recursion

Daily Time: 2-3 hours

Learning Objectives:

- Master recursive thinking and problem decomposition
- Understand tail recursion and optimization
- Build intuition for recursive solutions

Daily Schedule:

- **Monday-Tuesday**:
  - Advanced recursion concepts
  - Tail recursion and applications
  - Writing effective base cases
- **Wednesday-Thursday**:
  - All recursion practice problems from course
  - Print patterns, sum calculations using recursion
- **Friday**:
  - Additional recursion problems (Fibonacci variants, tree recursion)
  - Complex recursive algorithms
- **Saturday**:
  - Recursion speed practice - solve 10 problems in 3 hours
  - Focus on pattern recognition
- **Sunday**:
  - Review weak areas in recursion

- Solve LeetCode recursion medium problems

Key Topics: Recursion, Tail Recursion, Recursive Problem Solving

---

Week 3: Lists & Strings

Daily Time: 2-3 hours

Learning Objectives:

- Master array operations and manipulations
- Understand string algorithms and pattern matching
- Learn sliding window and two-pointer techniques

Daily Schedule:

- **Monday-Tuesday**:
  - List operations, comprehensions, slicing
  - Array algorithms (largest element, second largest, sorting check)
- **Wednesday**:
  - Advanced list problems (Leaders, rotate, move zeros)
  - Frequency calculations and duplicate removal
- **Thursday-Friday**:
  - String algorithms (anagram, palindrome, subsequence)
  - String rotations and pattern searching basics
- **Saturday**:
  - Mixed practice - 15 array/string problems
  - Speed building exercises
- **Sunday**:
  - LeetCode sliding window problems
  - Two-pointer technique mastery

Key Topics: Arrays, Strings, Sliding Window, Two Pointers

Week 4: Searching & Sorting

Daily Time: 2-3 hours

Learning Objectives:

- Master binary search and its variations
- Understand sorting algorithms and their complexities
- Learn when to apply different search/sort techniques

Daily Schedule:

- **Monday**:
  - Binary search implementation and analysis
  - First/last occurrence in sorted arrays
- **Tuesday**:
  - Sorting algorithms (bubble, selection, insertion)
  - Merge sort and quick sort implementation
- **Wednesday-Thursday**:
  - Advanced sorting algorithms and analysis
  - Heap sort and counting sort
- **Friday**:
  - Searching problems practice
  - Search in rotated arrays
- **Saturday**:
  - Sorting problems and complexity analysis
  - Stability in sorting algorithms
- **Sunday**:
  - Mixed practice problems
  - LeetCode search/sort medium problems

Key Topics: Binary Search, Sorting Algorithms, Search Variations

---

PHASE 2: CORE DATA STRUCTURES (Weeks 5-10)

Week 5: Hashing

Daily Time: 2-3 hours

Learning Objectives:

- Understand hash functions and collision handling
- Master Python dictionaries and sets
- Learn hash-based problem solving patterns

Daily Schedule:

- **Monday-Tuesday**:
  - Hashing theory, hash functions
  - Collision handling (chaining vs open addressing)
- **Wednesday**:
  - Python sets and dictionaries mastery
  - Implementation details and performance
- **Thursday-Friday**:
  - Hashing problems (distinct elements, frequency counting)
  - Subarray problems using hashing
- **Saturday**:
  - 20 hashing problems from course + LeetCode
  - Pattern recognition practice
- **Sunday**:
  - Hash map design patterns
  - Advanced hashing applications

Key Topics: Hash Tables, Sets, Dictionaries, Hash-based Algorithms

---

Week 6: Linked Lists

Daily Time: 2-3 hours

Learning Objectives:

- Master all types of linked lists
- Understand pointer manipulation techniques
- Learn fast/slow pointer patterns

Daily Schedule:

- **Monday**:
  - Singly linked list implementation

    - Basic operations (insert, delete, search)
- **Tuesday**:
    - Advanced operations (reverse, find middle, nth from end)
    - Remove duplicates and detect patterns
- **Wednesday**:
    - Circular linked lists
    - Implementation and applications
- **Thursday**:
    - Doubly linked lists
    - Bidirectional traversal and operations
- **Friday**:
    - All linked list problems from course
    - Complex manipulations and algorithms
- **Saturday**:
    - LeetCode linked list medium problems
    - Speed and accuracy practice
- **Sunday**:
    - Fast/slow pointer technique mastery
    - Cycle detection and intersection problems

Key Topics: Singly/Doubly/Circular Linked Lists, Pointer Manipulation

---

Week 7: Stacks & Queues

Daily Time: 2-3 hours

Learning Objectives:

- Understand LIFO and FIFO principles
- Master stack and queue applications
- Learn monotonic stack patterns

Daily Schedule:

- **Monday-Tuesday**:
    - Stack implementation and applications

    - Array and linked list based implementations
- **Wednesday**:
    - Queue and Deque implementation
    - Circular queue concepts
- **Thursday**:
    - Stack problems (balanced parentheses, next greater element)
    - Expression evaluation and conversion
- **Friday**:
    - Queue problems and BFS preparation
    - Priority queue concepts
- **Saturday**:
    - 15 stack/queue problems
    - Mixed applications practice
- **Sunday**:
    - Monotonic stack pattern practice
    - Advanced stack/queue algorithms

Key Topics: Stacks, Queues, Deques, Expression Evaluation

Week 8: Trees (Basic)

Daily Time: 3 hours

Learning Objectives:

- Understand tree data structure fundamentals
- Master tree traversal techniques
- Learn basic tree algorithms

Daily Schedule:

- **Monday-Tuesday**:
    - Binary tree structure and representation
    - All traversal methods (inorder, preorder, postorder)
- **Wednesday**:
    - Tree properties (height, size, level order traversal)

    - Iterative traversal implementations
- **Thursday**:
    - Tree problems from course
    - Distance and path algorithms
- **Friday**:
    - Binary Search Tree basics
    - Search, insert, delete operations
- **Saturday**:
    - 20 tree problems practice
    - Speed building for tree algorithms
- **Sunday**:
    - Tree pattern recognition
    - LeetCode tree problems

Key Topics: Binary Trees, Tree Traversals, Basic Tree Algorithms

---

Week 9: Trees (Advanced) & Heaps

Daily Time: 3 hours

Learning Objectives:

- Master advanced tree operations
- Understand heap data structure
- Learn priority queue applications

Daily Schedule:

- **Monday**:
    - BST advanced operations (floor, ceiling)
    - BST validation and fixing
- **Tuesday**:
    - Self-balancing trees (AVL, Red-Black theory)
    - Tree balancing concepts
- **Wednesday-Thursday**:
    - Heap implementation and heap sort

    - Min/max heap operations
- **Friday**:
    - Heap problems and heapq library mastery
    - Priority queue applications
- **Saturday**:
    - Advanced tree and heap problems
    - Complex tree algorithms
- **Sunday**:
    - Priority queue design patterns
    - Heap-based algorithms

Key Topics: BST Operations, Heaps, Priority Queues, Self-balancing Trees

---

Week 10: Bit Magic

Daily Time: 2-3 hours

Learning Objectives:

- Master bitwise operations
- Learn bit manipulation techniques
- Understand space-efficient algorithms

Daily Schedule:

- **Monday-Tuesday**:
    - Bitwise operations mastery (AND, OR, XOR, shifts)
    - Binary representation and bit patterns
- **Wednesday**:
    - Bit manipulation problems
    - Check/set/clear bit operations
- **Thursday-Friday**:
    - Advanced problems (power of 2, odd occurring elements)
    - Power set generation using bits
- **Saturday**:
    - 15 bit manipulation problems

- Optimization using bit operations

- **Sunday**:
  - Optimize previous solutions using bit tricks
  - Space-efficient algorithms

Key Topics: Bitwise Operations, Bit Manipulation, Space Optimization

---

PHASE 3: ADVANCED ALGORITHMS (Weeks 11-16)

Week 11: Advanced Arrays & Strings

Daily Time: 3 hours

Learning Objectives:

- Master advanced array techniques
- Learn string pattern matching algorithms
- Understand sliding window variations

Daily Schedule:

- **Monday**:
  - Sliding window technique mastery
  - Variable and fixed window problems
- **Tuesday**:
  - Kadane's algorithm and variations
  - Maximum subarray and circular array problems
- **Wednesday**:
  - String pattern matching algorithms
  - Naive pattern searching improvements
- **Thursday**:
  - KMP algorithm implementation
  - Rabin-Karp algorithm and rolling hash
- **Friday**:
  - Advanced string problems
  - Lexicographic problems and string transformations

- **Saturday**:
  - 25 advanced array/string problems
  - Speed and pattern recognition
- **Sunday**:
  - Pattern matching algorithm practice
  - Complex string manipulations

Key Topics: Advanced Arrays, String Algorithms, Pattern Matching

---

Week 12: Advanced Linked Lists & Stacks

Daily Time: 3 hours

Learning Objectives:

- Master complex linked list algorithms
- Learn advanced stack applications
- Understand design patterns with data structures

Daily Schedule:

- **Monday**:
  - Cycle detection algorithms (Floyd's algorithm)
  - Intersection point of linked lists
- **Tuesday**:
  - Clone linked list with random pointers
  - LRU cache design and implementation
- **Wednesday**:
  - Advanced stack problems (largest rectangle in histogram)
  - Stack with getMin() in O(1)
- **Thursday**:
  - Expression evaluation (infix, postfix, prefix)
  - Expression conversion algorithms
- **Friday**:
  - Implement stack using queues
  - Implement queue using stacks

- **Saturday**:
  - 20 advanced linked list and stack problems
  - Complex algorithm implementations
- **Sunday**:
  - Design pattern practice
  - System design using basic data structures

Key Topics: Advanced Linked Lists, Stack Applications, Data Structure Design

---

Week 13: Graphs (Part 1)

Daily Time: 3-4 hours

Learning Objectives:

- Understand graph representations
- Master graph traversal algorithms
- Learn basic graph problems

Daily Schedule:

- **Monday**:
  - Graph representation (adjacency matrix/list)
  - BFS and DFS implementation
- **Tuesday**:
  - Connected components using BFS/DFS
  - Cycle detection in undirected graphs
- **Wednesday**:
  - Topological sorting (Kahn's algorithm, DFS-based)
  - Cycle detection in directed graphs
- **Thursday**:
  - Shortest path in unweighted graphs
  - BFS applications
- **Friday**:
  - All basic graph problems from course
  - Graph property calculations

- **Saturday**:
  - Graph traversal practice - 15 problems
  - Implementation from scratch
- **Sunday**:
  - Graph pattern recognition
  - Problem classification techniques

Key Topics: Graph Traversal, Cycle Detection, Topological Sorting

---

Week 14: Graphs (Part 2)

Daily Time: 3-4 hours

Learning Objectives:

- Master shortest path algorithms
- Understand minimum spanning trees
- Learn advanced graph algorithms

Daily Schedule:

- **Monday**:
  - Dijkstra's algorithm implementation
  - Single source shortest path
- **Tuesday**:
  - Bellman-Ford algorithm
  - Negative cycle detection
- **Wednesday**:
  - Minimum Spanning Tree (Prim's and Kruskal's)
  - Disjoint set data structure
- **Thursday**:
  - Advanced graph algorithms (Kosaraju's, Tarjan's)
  - Strongly connected components
- **Friday**:
  - Graph algorithm problems practice
  - Real-world applications

- **Saturday**:
  - 20 graph algorithm problems
  - Mixed difficulty practice
- **Sunday**:
  - Graph optimization problems
  - Network flow basics

Key Topics: Shortest Path, MST, Advanced Graph Algorithms

Week 15: Dynamic Programming

Daily Time: 3-4 hours

Learning Objectives:

- Master dynamic programming concepts
- Learn memoization vs tabulation
- Understand DP patterns and variations

Daily Schedule:

- **Monday**:
  - DP concepts, memoization vs tabulation
  - Optimal substructure and overlapping subproblems
- **Tuesday**:
  - Longest Common Subsequence and variations
  - String-based DP problems
- **Wednesday**:
  - Longest Increasing Subsequence
  - Coin change problems (combinations and minimum coins)
- **Thursday**:
  - 0-1 Knapsack problem and variations
  - Subset sum problems
- **Friday**:
  - Matrix chain multiplication
  - Palindrome partitioning problems

- **Saturday**:
  - 15 classic DP problems
  - Pattern recognition in DP
- **Sunday**:
  - DP optimization techniques
  - Space optimization in DP

Key Topics: Dynamic Programming, Memoization, DP Patterns

Week 16: Advanced Topics & Integration

Daily Time: 3-4 hours

Learning Objectives:

- Learn remaining advanced topics
- Integrate all concepts learned
- Prepare for interviews and competitions

Daily Schedule:

- **Monday**:
  - Greedy algorithms and activity selection
  - Backtracking (N-Queens, Sudoku, Rat in Maze)
- **Tuesday**:
  - Trie data structure implementation
  - Trie applications and string problems
- **Wednesday**:
  - Segment trees and range queries
  - Binary Indexed Trees (Fenwick trees)
- **Thursday**:
  - Disjoint set union with path compression
  - Advanced data structure applications
- **Friday**:
  - Mixed practice covering all topics
  - Problem solving strategies

- **Saturday**:
  - 30 mixed problems (interview simulation)
  - Time pressure practice
- **Sunday**:
  - Week area revision
  - Speed practice and final review

Key Topics: Greedy, Backtracking, Tries, Segment Trees, Advanced Data Structures

---

DAILY STRUCTURE BREAKDOWN

Standard 2-4 Hour Daily Schedule:

Hour 1: Learning Phase (45-60 minutes)

- Watch lectures or read theory
- Understand concepts and algorithms
- Take notes on key patterns

Hour 2: Implementation Phase (45-60 minutes)

- Code algorithms from scratch
- Understand implementation details
- Debug and optimize code

Hour 3: Practice Phase (45-60 minutes)

- Solve course practice problems
- Apply learned concepts
- Build problem-solving speed

Hour 4: Advanced Practice (45-60 minutes)

- LeetCode problems related to topic
- Speed practice with timer
- Pattern recognition exercises

WEEKLY GOALS AND MILESTONES

Problem Solving Targets:

- **Weeks 1-4**: 50-60 problems per week
- **Weeks 5-10**: 60-70 problems per week
- **Weeks 11-16**: 70-80 problems per week

Speed Benchmarks:

- **Week 4**: Easy problems in 15-20 minutes
- **Week 8**: Medium problems in 30-45 minutes
- **Week 12**: Pattern recognition within 2-3 minutes
- **Week 16**: Hard problems systematic approach in 60+ minutes

Implementation Goals:

- **Weekly**: Implement 2-3 algorithms completely from scratch
- **Bi-weekly**: Create one data structure implementation
- **Monthly**: Build a mini-project using learned concepts

SUCCESS METRICS AND EVALUATION

Weekly Assessment:

- **Concept Understanding**: Can explain algorithm to someone else
- **Implementation Skill**: Can code algorithm without reference
- **Problem Solving**: Can solve related problems independently
- **Speed**: Meets weekly speed benchmarks

Monthly Milestones:

- **Month 1**: Strong foundation in basics and recursion
- **Month 2**: Proficiency in core data structures
- **Month 3**: Competency in graph algorithms and DP
- **Month 4**: Interview-ready with advanced topics mastery

Final Competency Markers:

- Solve 80% of LeetCode Medium problems
- Implement any basic algorithm from memory

- Recognize patterns in new problems quickly
- Explain time/space complexity for any solution
- Design efficient solutions for novel problems

RECOMMENDED RESOURCES

Primary:

- GeeksforGeeks DSA Course (provided curriculum)
- LeetCode for additional practice
- Personal coding notebook for patterns

Supplementary:

- "Cracking the Coding Interview" by Gayle McDowell
- LeetCode discuss section for alternative solutions
- Visualgo.net for algorithm visualizations

Tools:

- Python IDE (PyCharm, VS Code)
- Timer for speed practice
- Notebook for pattern documentation
- LeetCode premium for company-specific problems

QUICK REFERENCE TABLE

Week	Phase	Focus Area	Daily Hours	Key Topics	Problems/Week
1	Foundation	Analysis & Math	2-3	Time Complexity, Recursion Basics	50-60
2	Foundation	Deep Recursion	2-3	Advanced Recursion, Tail Recursion	50-60
3	Foundation	Arrays & Strings	2-3	Sliding Window, Two Pointers	50-60
4	Foundation	Search & Sort	2-3	Binary Search, Sorting Algorithms	50-60
5	Core DS	Hashing	2-3	Hash Tables, Sets, Dictionaries	60-70
6	Core DS	Linked Lists	2-3	All LL Types, Pointer Manipulation	60-70
7	Core DS	Stacks & Queues	2-3	LIFO/FIFO, Expression Evaluation	60-70
8	Core DS	Trees Basic	3	Binary Trees, Traversals, BST	60-70
9	Core DS	Trees Advanced & Heaps	3	Advanced BST, Heaps, Priority Queues	60-70
10	Core DS	Bit Magic	2-3	Bitwise Operations, Bit Manipulation	60-70
11	Advanced	Advanced Arrays/Strings	3	Advanced Techniques, Pattern Matching	70-80
12	Advanced	Advanced DS Applications	3	Complex Algorithms, Design Patterns	70-80
13	Advanced	Graphs Part 1	3-4	Traversal, Cycles, Topological Sort	70-80
14	Advanced	Graphs Part 2	3-4	Shortest Path, MST, Advanced Graphs	70-80
15	Advanced	Dynamic Programming	3-4	DP Patterns, Memoization, Optimization	70-80
16	Advanced	Integration & Advanced Topics	3-4	Greedy, Backtracking, Tries, Review	70-80

Total Duration: 16 Weeks (4 Months) **Total Time Investment**: 350-400 hours **Expected Outcome**: Interview-ready with strong DSA foundation **Problem Solving Count**: 1000+ problems across all difficulty levels