7-Day Plan to Master Data Structures and Algorithms with Java

This plan balances theoretical concepts and practical coding exercises, ensuring mastery of fundamental and advanced topics.

Day 1: Basics of Data Structures and Complexity Analysis

Topics to Cover:

- Introduction to Data Structures.
- Big-O Notation:
  - Time Complexity (O(1), O(n), O(n²)).
  - Space Complexity.
- Arrays:
  - Static arrays in Java.
  - Basic operations: insertion, deletion, traversal.

Practical Task:

1. Write a Java program to:
   - Implement basic operations on an array (e.g., adding/removing elements).
   - Find the largest and smallest elements in an array.
2. Analyze the time complexity of your implementations.

Resources:

- Big-O Cheatsheet
- Java Arrays Documentation

Day 2: Strings and Recursion

Topics to Cover:

- Strings:
  - String manipulation in Java (`StringBuilder` , `StringBuffer`).
- Recursion:
  - Concept and use cases.

    - Writing recursive functions.
    - Solving problems with recursion.

Practical Task:

1. Write a Java program to:
    - Reverse a string without using built-in functions.
    - Check if a string is a palindrome.
2. Implement a recursive solution for:
    - Calculating the factorial of a number.
    - Generating Fibonacci numbers.

Resources:

- Java Strings

Day 3: Linked Lists

Topics to Cover:

- Singly Linked List:
    - Insertion, Deletion, Traversal.
- Doubly Linked List:
    - Forward and backward traversal.
- Comparison between arrays and linked lists.

Practical Task:

1. Implement a singly linked list in Java:
    - Add, remove, and search for elements.
2. Extend your implementation to include a doubly linked list.

Resources:

- Linked Lists in Java

Day 4: Stacks and Queues

Topics to Cover:

- Stacks:
  - Operations: `push` , `pop` , `peek` .
  - Implementation using arrays and linked lists.
- Queues:
  - Operations: `enqueue` , `dequeue` .
  - Types: Simple Queue, Circular Queue, Priority Queue.
- Applications:
  - Balancing parentheses, undo operations in text editors.

Practical Task:

1. Write a Java program to:
   - Implement a stack and queue.
   - Check for balanced parentheses using a stack.
   - Implement a circular queue.

Resources:

- Java Stack Class
- Queue in Java

Day 5: Trees and Binary Search Trees

Topics to Cover:

- Binary Trees:
  - Concepts, traversal methods (in-order, pre-order, post-order).
- Binary Search Trees (BST):
  - Insertion, Deletion, Search.
- Applications of BST.

Practical Task:

1. Implement a binary tree in Java:
   - Perform all types of traversals.
2. Extend the tree to a binary search tree:

    - Insert elements and search for a value.

Resources:

- Binary Trees

Day 6: Sorting and Searching Algorithms

Topics to Cover:

- Sorting:
    - Bubble Sort, Selection Sort, Insertion Sort.
    - Advanced Sorting: Merge Sort, Quick Sort.
- Searching:
    - Linear Search, Binary Search.
- Analysis of time and space complexity for each algorithm.

Practical Task:

1. Write Java programs to:
    - Implement and test all sorting algorithms.
    - Perform binary search on a sorted array.

Resources:

- Sorting Algorithms

Day 7: Graphs and Hashing

Topics to Cover:

- Graph Basics:
    - Representation: Adjacency Matrix, Adjacency List.
    - Graph Traversal: BFS, DFS.
- Hashing:
    - HashMaps in Java.
    - Collision Handling: Chaining, Open Addressing.

Practical Task:

1. Write Java programs to:
    - Represent a graph using adjacency lists.
    - Perform BFS and DFS on a graph.
    - Use a `HashMap` to count the frequency of elements in an array.

Resources:

- Graphs in Java
- HashMap Class

Bonus (Optional Advanced Topics)

- Dynamic Programming:
    - Solve problems like the knapsack problem, longest common subsequence.
- Advanced Trees:
    - AVL Trees, Red-Black Trees, Tries.
- Heaps:
    - Min-Heap, Max-Heap.

This plan provides a solid foundation in data structures and algorithms, with practical exercises to reinforce learning. Let me know if you’d like further elaboration or coding examples for specific topics!