**Roadmap for DSA to Crack High-Paying Jobs (For Java Beginners)**

**Stage 1: Basics of Java Programming - Done**

Before diving into DSA, you need a strong foundation in Java:

1. **Java Fundamentals:**
   * Java Setup (JDK, IDE - IntelliJ/Eclipse)
   * Variables and Data Types
   * Operators and Expressions
   * Control Flow (if-else, switch-case)
   * Loops (for, while, do-while)
   * Functions/Methods
   * Input and Output (Scanner class)
2. **Object-Oriented Programming Concepts:**
   * Classes and Objects
   * Constructors
   * Inheritance
   * Polymorphism (method overloading and overriding)
   * Encapsulation
   * Abstraction
   * Interfaces and Abstract classes
3. **Basic Java Utilities:**
   * Arrays
   * String Handling
   * Exception Handling
   * Collections Framework (ArrayList, LinkedList, HashMap basics)

**Stage 2: Understanding Data Structures (1/7 - Done)**

Learn about each data structure, how it works, and how to implement it in Java:

**1. Arrays and Strings**

* One-dimensional and Multi-dimensional arrays
* Basic operations: insertion, deletion, traversal, searching
* Common problems: Reverse, Rotate, Max/Min, Frequency
* String manipulation, StringBuilder, StringBuffer

**2. Linked Lists**

* Singly Linked List
* Doubly Linked List
* Circular Linked List
* Operations: insertion, deletion, traversal, searching
* Common problems: Detect cycle, Reverse Linked List, Merge Two Lists

**3. Stacks**

* Concept & Applications
* Implement using arrays and linked list
* Common problems: Balanced parentheses, Next Greater Element, Infix/Postfix/Prefix evaluation

**4. Queues**

* Simple Queue, Circular Queue
* Priority Queue
* Deque (Double-ended Queue)
* Implementations using arrays and linked lists
* Common problems: Sliding Window Maximum, BFS traversal in Graphs

**5. Trees**

* Binary Trees
* Binary Search Trees (BST)
* Tree traversals (Inorder, Preorder, Postorder)
* Height/Depth of a tree
* Balanced Trees (AVL, Red-Black Trees - basic understanding)
* Common problems: Lowest Common Ancestor, Tree Diameter, Serialize/Deserialize Tree

**6. Graphs**

* Graph representation: Adjacency matrix/list
* Types of graphs (directed, undirected, weighted)
* Graph traversal: BFS, DFS
* Shortest path algorithms (Dijkstra’s Algorithm - basic)
* Detect cycle in graph
* Connected components

**7. Hashing**

* HashMap and HashSet in Java
* Handling collisions (Chaining, Open Addressing)
* Applications: Frequency count, Anagrams, Subarray sum problems

**Stage 3: Algorithms**

Learn algorithms in detail and understand their time complexities:

**1. Sorting Algorithms**

* Bubble Sort, Selection Sort, Insertion Sort
* Merge Sort (Divide & Conquer)
* Quick Sort
* Counting Sort, Radix Sort, Bucket Sort (basic understanding)
* Understand Big O complexity for all

**2. Searching Algorithms**

* Linear Search
* Binary Search (Iterative and Recursive)
* Search in Rotated Sorted Array
* Search in 2D Matrix

**3. Recursion and Backtracking**

* Understand recursion basics and stack frames
* Common problems: Factorial, Fibonacci, Tower of Hanoi
* Backtracking basics
* Problems: N-Queens, Sudoku Solver, Subset/Permutation generation

**4. Dynamic Programming (DP)**

* Understand memoization and tabulation
* Classic problems:
  + Fibonacci with DP
  + Knapsack Problem (0/1 and Unbounded)
  + Longest Common Subsequence (LCS)
  + Coin Change
  + Edit Distance
  + Matrix Chain Multiplication

**5. Greedy Algorithms**

* Concept and comparison with DP
* Problems: Activity Selection, Huffman Encoding, Fractional Knapsack

**Stage 4: Problem Solving Practice**

* Start with easy problems on platforms like LeetCode, CodeChef, HackerRank, or GeeksforGeeks.
* Progress to medium and then hard problems.
* Topics to focus on in practice:
  + Arrays and Strings problems
  + Linked List problems
  + Stack and Queue problems
  + Tree and Graph problems
  + DP and Backtracking problems

**Stage 5: Advanced Topics & Interview Preparation**

1. **Advanced Data Structures (optional but helpful)**
   * Trie
   * Segment Trees
   * Fenwick Tree (Binary Indexed Tree)
   * Disjoint Set Union (Union-Find)
2. **System Design Basics (for senior roles)**
3. **Mock Interviews and Coding Challenges**
   * Solve problems under time constraints
   * Participate in contests on Codeforces, AtCoder
   * Practice explaining your thought process clearly

**Stage 6: Resume and Interview Strategy**

* Build projects or contribute to open source using Java.
* Prepare for behavioral questions.
* Review common interview patterns.
* Practice whiteboard coding or using online coding platforms.

**Summary Timeline Suggestion**

| **Week(s)** | **Focus Area** |
| --- | --- |
| 1-3 | Java Basics and OOP |
| 4-6 | Arrays, Strings, and Basic Data Structures (Linked Lists, Stack, Queue) |
| 7-9 | Trees and Graphs |
| 10-12 | Sorting, Searching, Recursion |
| 13-15 | Backtracking and Dynamic Programming |
| 16-18 | Advanced algorithms + Problem Solving |
| 19-22 | Mock Interviews + System Design basics |