| **Week** | **Day** | **Topics** | **Details** | **Practice** |
| --- | --- | --- | --- | --- |
| **Week 1** | **Day 1-2** | Java Fundamentals Review | - **Arrays, Strings**, basic data types, OOP concepts: Classes, Objects, Inheritance, Polymorphism, etc. | - Write small programs for practice (loops, conditions, arrays) |
|  | **Day 3-5** | Arrays & Strings | - Array manipulation (searching, sorting, reversing, two-pointer technique, etc.)- String operations (anagrams, substrings, palindromes, etc.) | - 10-15 problems from LeetCode or HackerRank (easy-medium) |
|  | **Day 6-7** | Time Complexity & Space Complexity | - Understand Big-O notation, time complexity, space complexity.- Best, worst, and average case analysis | - Solve 5-7 problems focusing on  time-space complexity analysis |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week 2** | **Day 8-9** | Linked Lists | - Single Linked List (Insertion, Deletion, Reversal, Cycle detection)- Doubly Linked List | - 5-10 problems on Linked Lists from LeetCode/HackerRank |
|  | **Day 10-11** | Stacks & Queues | - Stack (Infix, Prefix, Postfix evaluations, Balanced parentheses)- Queue (Circular Queue, Priority Queue) | - 5-10 problems (medium difficulty) |
|  | **Day 12-13** | Trees | - Binary Tree Traversals (Inorder, Preorder, Postorder)- DFS, BFS- Binary Search Tree (BST), LCA, Path Sum | - 5-10 tree traversal problems |
|  | **Day 14** | Heaps & HashMaps | - Heaps (Max-Heap, Min-Heap, PriorityQueue)- HashMap (frequency counting, two-sum problem) | - 5-7 problems related to heaps and hashmaps |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week 3** | **Day 15-16** | Graphs | - Graph Representations (Adjacency List, Matrix)- BFS, DFS, Dijkstra’s Algorithm- Topological Sort, Cycle detection | - 5-7 problems involving graph traversal, shortest path |
|  | **Day 17-18** | Recursion & Backtracking | - Understand recursion, basic backtracking problems (N-Queens, Subset Sum, Permutations) | - 5-7 backtracking problems (medium difficulty) |
|  | **Day 19-20** | Dynamic Programming (DP) | - DP Techniques: Memoization, Tabulation- Classic problems (Fibonacci, LCS, Knapsack, Coin Change, LIS) | - 5-7 dynamic programming problems |
|  | **Day 21** | Greedy Algorithms | - Understand Greedy approach for problems like Activity Selection, Fractional Knapsack, Job Scheduling | - 5-7 greedy problems |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week 4** | **Day 22-23** | Problem Solving with Time Constraints | - Solve 10-15 problems under timed conditions (45-60 minutes per problem) | - Practice on LeetCode/CodeForces with time limits |
|  | **Day 24-25** | Mock Interviews | - Conduct mock interviews with a peer, mentor, or online platform (Pramp, Interviewing.io, LeetCode mock interviews) | - Complete at least 2-3 mock interviews |
|  | **Day 26-28** | Review Weak Areas | - Revisit difficult topics and concepts.- Revise key algorithms (sorting, searching, binary search). | - Practice problems in areas of weakness (LeetCode, HackerRank) |
|  | **Day 29-30** | Final Revision & Rest | - Final review of core concepts.- Practice 1-2 problems you are confident about.- Rest, relax, and mentally prepare. | - Light practice, no heavy coding |

**Key Guidelines to Follow:**

* **Consistency**: Dedicate 4-5 hours per day, ensuring consistent practice and revision.
* **Understanding Concepts**: Focus on understanding **algorithm design** rather than just memorizing solutions.
* **Practice Timed Problems**: On **Day 22-23**, practice solving problems under time constraints. This simulates real interview pressure.
* **Mock Interviews**: On **Day 24-25**, do mock interviews to simulate a real interview experience and get feedback.
* **Focus on Weaknesses**: In **Week 4**, revisit any weak areas, whether it’s a specific data structure or a particular algorithm.
* **Review Common Patterns**: Recognize common interview problem-solving patterns (sliding window, two-pointer, recursion, dynamic programming, etc.).

**Additional Tips:**

* **Use Java’s built-in libraries**: Get comfortable using ArrayList, HashMap, PriorityQueue, and other Java collections to speed up coding.
* **Time and Space Complexity**: Always analyze your solution for time and space efficiency.
* **Communication Skills**: In interviews, articulate your thought process clearly and ask questions if the problem statement is ambiguous.