DATA STRUCTURES AND ALGORITHMS (DSA) – CONTENT

- Phase 7.

Phase 1: Introduction and Fundamentals

1. **What Are Data Structures and Algorithms?**
   Introduce DSA, its purpose, and its importance in programming and real-world applications.
2. **Why Learn DSA? Applications in Real Life**
   Showcase practical applications of DSA, motivating viewers to learn.
3. **Time Complexity and Big-O Notation**
   Explain the importance of analyzing algorithm efficiency with real-world examples.
4. **Space Complexity: Why It Matters**
   Dive into how space usage impacts program performance.
5. **How to Analyze an Algorithm**
   Discuss key steps in understanding and evaluating algorithms.

Phase 2: Arrays and Strings

6. **Introduction to Arrays**
   Cover the basics and why arrays are foundational in programming.
7. **Array Operations: Insert, Delete, Update**
   Teach common operations step-by-step with coding examples.
8. **Two-Dimensional Arrays and Matrix Representation**
   Introduce 2D arrays and their use cases like games, grids, and images.
9. **Applications of Arrays: Examples in Real Life**
   Relate arrays to real-world scenarios, such as managing data in spreadsheets.
10. **Introduction to Strings in Programming**
    Explain strings as a data structure and their importance in software development.
11. **String Operations: Reverse, Concatenate, and More**
    Show common string manipulations with coding demonstrations.
12. **Pattern Matching Algorithms (Naive, KMP, etc.)**
    Dive into pattern matching with examples like searching text or DNA sequences.
13. **Practical Uses of Strings: Spell Checkers, DNA Matching**
    Highlight real-world applications to cement understanding.

Phase 3: Linked Lists

14. **Introduction to Linked Lists: Basics and Types**
    Explain the concept, advantages, and types of linked lists.
15. **Singly Linked List: Implementation and Operations**
    Code basic operations like insertion, deletion, and traversal.
16. **Doubly Linked List: Implementation and Operations**
    Show how to handle bidirectional traversal and additional flexibility.

17. **Circular Linked List: Basics and Use Cases**
    Cover special cases and their relevance in scenarios like buffering.
18. **Comparing Linked Lists and Arrays**
    Discuss when to use linked lists over arrays and vice versa.
19. **Applications of Linked Lists: Real-World Examples**
    Provide practical examples, such as memory management or undo features.

Phase 4: Stacks and Queues

20. **What Is a Stack? LIFO Explained**
    Introduce stacks and their use in programming problems.
21. **Stack Operations: Push, Pop, Peek**
    Demonstrate operations step-by-step with coding examples.
22. **Applications of Stacks: Function Calls, Undo/Redo**
    Explain how stacks work behind the scenes in common use cases.
23. **Implementing a Stack Using Arrays**
    Implement stacks using arrays for a practical approach.
24. **Implementing a Stack Using Linked Lists**
    Compare and contrast with the linked list implementation.
25. **Problems Solved by Stacks: Parentheses Matching, Next Greater Element**
    Solve popular stack-related problems with code walkthroughs.
26. **What Is a Queue? FIFO Explained**
    Introduce queues with simple examples like waiting lines.
27. **Queue Operations: Enqueue, Dequeue, Peek**
    Walk through coding operations.
28. **Circular Queue: Implementation and Benefits**
    Explain circular queues for efficient space usage.
29. **Priority Queue: Basics and Applications**
    Dive into prioritized task handling with examples like job scheduling.

30. **Double-Ended Queue (Deque): Implementation**
    Show the flexibility of deques for double-ended operations.
31. **Real-World Applications of Queues: Task Scheduling**
    Explain how queues power scheduling, customer service, etc.

Phase 5: Trees

32. **Introduction to Trees and Terminology**
    Explain basic tree structures, nodes, edges, and levels.
33. **Binary Trees: Basics and Traversals (Inorder, Preorder, Postorder)**
    Demonstrate traversal techniques with code.
34. **Binary Search Tree (BST): Implementation and Operations**
    Implement BSTs with search, insert, and delete operations.
35. **AVL Trees: Self-Balancing Trees**
    Introduce balancing for optimal performance.
36. **Red-Black Trees: Balancing and Applications**
    Cover another balancing tree with real-world use cases.
37. **Heap Data Structure: Min-Heap and Max-Heap**
    Explain heaps and their use in priority queues and sorting.
38. **Applications of Trees: File Systems, Expression Parsing**
    Highlight the role of trees in organizing hierarchical data.

Phase 6: Graphs

39. **Introduction to Graphs: Nodes and Edges**
    Cover the fundamentals of graphs and their representations.
40. **Types of Graphs: Directed, Undirected, Weighted**
    Explain variations and their real-world examples.

41. **Graph Representations: Adjacency Matrix and List**
    Show how graphs are stored in memory.
42. **Breadth-First Search (BFS): Algorithm and Applications**
    Explain BFS and its applications like shortest path finding.
43. **Depth-First Search (DFS): Algorithm and Applications**
    Compare with BFS and explore applications like cycle detection.
44. **Topological Sorting: Applications in Task Scheduling**
    Show its role in ordering tasks with dependencies.
45. **Dijkstra's Algorithm: Finding Shortest Paths**
    Explain shortest-path calculation for weighted graphs.
46. **Bellman-Ford Algorithm: Handling Negative Weights**
    Handle edge cases with negative weights.
47. **Floyd-Warshall Algorithm: All-Pairs Shortest Path**
    Discuss efficient computation of paths between all nodes.
48. **Minimum Spanning Trees: Prim's and Kruskal's Algorithms**
    Solve problems like network design with MSTs.

Phase 7: Advanced Topics

49. **Dynamic Programming**
    Cover popular problems like Knapsack, LCS, and LIS.

- **Knapsack Problem: Recursion vs. DP**
- **Longest Common Subsequence (LCS) Problem**
- **Longest Increasing Subsequence Problem**

50. **Greedy Algorithms**
    Solve problems like Huffman Encoding and Activity Selection.
51. **Backtracking**
    Tackle problems like N-Queens and Subset Sum.

52. **Divide and Conquer**
    Revisit concepts in sorting and searching.
53. **Hashing**
    Explain hashing, collision handling, and applications.
54. **Advanced Topics**

- Tries
- Sliding Window Technique
- Disjoint Set Union (DSU)

Final Phase: Wrap-Up

55. **How to Approach DSA Problems in Interviews**
    Tips and strategies for tackling interview questions.
56. **DSA Practice Platforms: Where to Hone Your Skills**
    Recommend platforms like LeetCode, HackerRank, etc.
57. **Recap and Cheat Sheet for DSA**
    Summarize key concepts and provide handy references.