**Data Structures and Algorithms in Python**

**1. Data Structures Basics**

**1.1 Analysis of Algorithms (Background)**

* Understand the importance and background of algorithm analysis.

**1.2 Asymptotic Analysis**

* Learn about asymptotic analysis and its significance.

**1.3 Order of Growth**

* Analyze how the runtime of an algorithm scales with input size.

**1.4 Best, Average, and Worst Cases**

* Understand the different scenarios of algorithm performance.

**1.5 Asymptotic Notations**

* Big O Notation
* Omega Notation
* Theta Notation

**1.6 Analysis of Common Loops**

* Examine time complexity for single, nested, and multiple loops.

**1.7 Analysis of Recursion**

* Recursion Tree Method
* Solving Recurrences
* Upper Bound using Recursion Tree Method

**1.8 Space Complexity**

* Measure the space required by an algorithm.

**Practice Problems**

* Implement basic algorithms and measure their time and space complexity.

**2. Mathematics for DSA**

**2.1 Count Digits**

**2.2 Palindrome Number**

**2.3 Factorial of a Number**

**2.4 Trailing Zeros in Factorial**

**2.5 GCD and HCF of Two Numbers**

**2.6 LCM of Two Numbers**

**2.7 Check for Prime**

**2.8 Prime Factors**

**2.9 All Divisors of a Number**

**2.10 Sieve of Eratosthenes**

**2.11 Computing Power**

**2.12 Iterative Power**

**Practice Problems**

* Solve mathematical problems using DSA concepts.

**3. List (Dynamic Sized Array)**

**3.1 Introduction to List**

**3.2 Working of List in Python**

**3.3 Average or Mean of a List**

**3.4 Separate Even and Odd**

**3.5 Get Smaller Elements**

**3.6 Slicing (List, Tuple, and String)**

**3.7 Comprehensions in Python**

**3.8 Largest Element in a List**

**3.9 Second Largest Element in a List**

**3.10 Check if a List is Sorted**

**3.11 Find the Only Odd**

**3.12 Reverse a List in Python**

**3.13 Remove Duplicates from Sorted Array**

**3.14 Move Zeros to End**

**3.15 Leaders in an Array Problem**

**3.16 Frequencies in a Sorted Array**

**3.17 Left Rotate a List by One**

**Practice Problems**

* Solve array and list-related problems to strengthen your understanding.

**4. Recursion**

**4.1 Applications of Recursion**

**4.2 Writing Base Cases in Recursion**

**4.3 Tail Recursion**

**4.4 Print N to 1 using Recursion**

**4.5 Print 1 to N using Recursion**

**4.6 Sum of Natural Numbers Using Recursion**

**4.7 Sum of Digits Using Recursion**

**4.8 Palindrome Check using Recursion**

**Practice Problems**

* Implement various recursive algorithms.

**5. Searching**

**5.1 Binary Search in Python**

**5.2 Recursive Binary Search in Python**

**5.3 Analysis of Binary Search**

**5.4 Index of First Occurrence in a Sorted Array**

**5.5 Index of Last Occurrence**

**5.6 Count Occurrences in a Sorted Array**

**5.7 Count 1s in a Sorted Binary Array**

**Practice Problems**

* Solve various searching-related problems.

**6. Sorting**

**6.1 Introduction to Sorting**

**6.2 List Sort in Python**

**6.3 Sorted in Python**

**6.4 Stability in Sorting Algorithms**

**6.5 Bubble Sort**

**6.6 Selection Sort**

**6.7 Insertion Sort in Python**

**6.8 Merge Sort Algorithm**

**6.9 Merge Two Sorted Arrays**

**6.10 Merge Subarrays**

**6.11 Count Inversions in Array**

**6.12 Merge Sort Analysis**

**6.13 Quick Sort Introduction**

**6.14 Partition a Given Array**

**6.15 Lomuto Partition**

**6.16 Hoare's Partition**

**6.17 Quick Sort using Lomuto Partition**

**6.18 Quick Sort using Hoare's Partition**

**6.19 Analysis of Quick Sort**

**6.20 Space Analysis of Quick Sort**

**6.21 Heap Sort**

**Practice Problems**

* Implement and analyze various sorting algorithms.

**7. Hashing**

**7.1 Introduction to Hashing**

**7.2 Hashing Applications**

**7.3 Direct Address Table**

**7.4 Hashing Functions**

**7.5 Collision Handling**

**7.6 Chaining**

**7.7 Implementation of Chaining in Python**

**7.8 Open Addressing**

**7.9 Double Hashing**

**7.10 Implementation of Open Addressing in Python**

**7.11 Chaining vs Open Addressing**

**7.12 Set in Python**

**7.13 Dictionary in Python**

**7.14 Count Distinct Elements in a List**

**7.15 Frequencies of Array Elements**

**Practice Problems**

* Solve hashing-related problems to understand different hashing techniques.

**8. Strings**

**8.1 Strings in Python**

**8.2 Escape Sequences and Raw Strings**

**8.3 Formatted String in Python**

**8.4 String Comparison in Python**

**8.5 String Operations Part (1)**

**8.6 String Operations Part (2)**

**8.7 Reverse a String in Python**

**8.8 Check if String is Rotated**

**8.9 Check for Palindrome in Python**

**8.10 Check if a String is a Subsequence of Another**

**8.11 Check for Anagram in Python**

**8.12 Leftmost Repeating Character**

**8.13 Leftmost Non-Repeating Element**

**8.14 Reverse Words in a String**

**Practice Problems**

* Solve string manipulation problems.

**9. Linked Lists**

**9.1 Problems with Array Data Structure**

**9.2 Linked List Introduction in Python**

**9.3 Simple Linked List Implementation in Python**

**9.4 Applications of Linked List**

**9.5 Traversing a Linked List in Python**

**9.6 Search in Linked List**

**9.7 Insert at the Beginning of Linked List in Python**

**9.8 Insert at the End of Linked List**

**9.9 Insert at a Given Position in Singly Linked List**

**9.10 Delete First Node of Linked List in Python**

**9.11 Delete Last Node of Linked List**

**9.12 Delete a Node with Pointer Given to It**

**9.13 Sorted Insert Linked List in Python**

**9.14 Middle of Linked List**

**9.15 Nth Node from End of Linked List**

**9.16 Remove Duplicates from a Sorted Singly Linked List**

**9.17 Reverse a Linked List in Python**

**9.18 Recursive Reverse a Linked List (Part 1)**

**9.19 Recursive Reverse a Linked List (Part 2)**

**Practice Problems**

* Solve linked list-related problems to understand their applications.

**10. Circular Linked List**

**10.1 Circular Linked List in Python**

**10.2 Circular Linked List (Advantages & Disadvantages)**

**10.3 Circular Linked List Traversal**

**10.4 Insert at the Beginning of Circular Linked List**

**10.5 Insert at the End of a Circular Linked List**

**10.6 Delete Head of Circular Linked List**

**10.7 Delete Kth Node of Circular Linked List**

**Practice Problems**

* Solve problems related to circular linked lists.

**11. Doubly Linked List**

**11.1 Doubly Linked List in Python**

**11.2 Singly vs Doubly Linked List (Advantages & Disadvantages)**

**11.3 Insert at the Beginning of DLL in Python**

**11.4 Insert at the End of DLL in Python**

**11.5 Delete Head of a Doubly Linked List**

**11.6 Delete Last Node of DLL in Python**

**11.7 Reverse a Doubly Linked List in Python**

**Practice Problems**

* Solve doubly linked list problems.

**12. Stacks**

**12.1 Stack Data Structure**

**12.2 Stack in Python**

**12.3 Linked List Implementation of Stack in Python**

**12.4 Stack Applications**

**12.5 Check for Balanced Parentheses in Python**

**Practice Problems**

* Implement various stack-related problems.

**13. Queues**

**13.1 Queue in Python**

**13.2 Queue Data Structure**

**13.3 Application of Queue Data Structure**

**13.4 Implementation of Queue using Array**

**13.5 Linked List Implementation of Queue in Python**

**Practice Problems**

* Solve problems related to queues.

**14. Deque**

**14.1 Deque Introduction**

**14.2 Deque Applications**

**14.3 Deque in Python**

**14.4 List Implementation of Deque in Python**

**14.5 Linked List Implementation of Deque**

**Practice Problems**

* Implement and solve problems using deques.

**15. Trees**

**15.1 Tree Data Structure**

**15.2 Application of Tree**

**15.3 Binary Tree in Python**

**15.4 Tree Traversal**

* Inorder Traversal in Python
* Preorder Traversal in Python
* Postorder Traversal in Python

**15.5 Height of Binary Tree**

**15.6 Print Node at K Distance**

**15.7 Level Order Traversal**

**15.8 Size of Binary Tree in Python**

**15.9 Maximum in Binary Tree**

**15.10 Iterative Inorder Traversal**

**15.11 Iterative Preorder Traversal**

**15.12 Iterative Preorder Traversal (Space Optimized)**

**Practice Problems**

* Solve tree traversal and manipulation problems.

**16. Binary Search Tree**

**16.1 Introduction to Binary Search Tree**

**16.2 Search in BST in Python**

**16.3 BST Insert in Python**

**16.4 BST Delete in Python**

**16.5 Floor in BST**

**16.6 Ceiling in BST**

**16.7 Self-Balancing BST**

* AVL Tree
* Red-Black Tree

**16.8 Applications of BST**

**Practice Problems**

* Implement BST operations and solve related problems.

**17. Heap**

**17.1 Binary Heap Introduction**

**17.2 Heap Python Implementation**

**17.3 Binary Heap Insert**

**17.4 Binary Heap Extract Min and Heapify**

**17.5 Decrease Key and Delete Operations**

**17.6 Build Heap**

**17.7 Heap Sort**

**17.8 Heapq in Python**

**Practice Problems**

* Solve heap-related problems and implement various heap operations.

**18. Bit Magic**

**18.1 Bitwise Operator in Python - Part 1**

**18.2 Bitwise Operator in Python - Part 2**

**18.3 Check Kth Bit is Set or Not**

**18.4 Count Set Bits**

**18.5 Power of Two**

**18.6 One Odd Occurring**

**18.7 Two Odd Occurring**

**18.8 Power Set Using Bitwise**

**Practice Problems**

* Solve problems using bitwise operations.

**19. Advanced Data Structures**

**19.1 Trie Data Structure**

**19.2 Trie Representation, Search, and Insert**

**19.3 Trie Delete**

**19.4 Count Distinct Rows in a Binary Matrix**

**19.5 Segment Tree**

* Introduction to Segment Tree
* Constructing Segment Tree
* Range Query on Segment Tree
* Update Query on Segment Tree

**19.6 Binary Indexed Tree**

* Introduction to Binary Indexed Tree
* Example Problem
* Prefix Sum
* Prefix Sum Implementation
* Update Operation

**Practice Problems**

* Implement and solve problems using advanced data structures.

**20. Disjoint Set**

**20.1 Disjoint Set Introduction**

**20.2 Find and Union Operations on Disjoint Sets**

**20.3 Union by Rank**

**20.4 Path Compression**

**20.5 Kruskal's Algorithm**

**Practice Problems**

* Implement and solve problems using disjoint sets.

**21. Graphs**

**21.1 Introduction to Graph**

**21.2 Graph Representation**

* Adjacency Matrix
* Adjacency List
* Adjacency List Representation in Python
* Comparison of Adjacency Matrix and List

**21.3 Breadth First Search (BFS) in Python**

* BFS for Disconnected Graph
* Connected Components in an Undirected Graph using BFS
* Applications of BFS

**21.4 Depth First Search (DFS)**

* DFS for Disconnected Graph
* Connected Components in an Undirected Graph using DFS
* Applications of DFS

**21.5 Shortest Path in an Unweighted Graph**

**21.6 Detect Cycle in Undirected Graph**

**21.7 Detect Cycle in a Directed Graph**

* Part 1
* Part 2

**21.8 Topological Sorting**

* Kahn's BFS Based Algorithm
* DFS Based Algorithm

**21.9 Shortest Path in DAG**

**21.10 Prim's Algorithm/Minimum Spanning Tree**

* Implementation of Prim's Algorithm

**21.11 Dijkstra's Shortest Path Algorithm**

* Implementation of Dijkstra's Algorithm

**21.12 Kosaraju's Algorithm**

* Part 1
* Part 2

**21.13 Bellman Ford Shortest Path Algorithm**

**21.14 Articulation Point**

**21.15 Bridges in Graph**

**21.16 Tarjans Algorithm**

**21.17 Kruskal's Algorithm**

**Practice Problems**

* Solve various graph-related problems.

**22. Greedy Algorithms**

**22.1 Introduction to Greedy Algorithms**

**22.2 Activity Selection Problem**

**22.3 Fractional Knapsack**

**22.4 Job Sequencing Problem**

**22.5 Huffman Coding**

* Introduction
* Huffman Algorithms
* Python Implementation of Huffman Coding

**Practice Problems**

* Implement and solve problems using greedy algorithms.

**23. Backtracking**

**23.1 Concept of Backtracking**

**23.2 Rat in a Maze**

**23.3 N Queen Problem**

**23.4 Sudoku Problem**

**Practice Problems**

* Solve problems using backtracking techniques.

**24. Dynamic Programming**

**24.1 Introduction to Dynamic Programming**

* Memoization
* Tabulation

**24.2 Longest Common Subsequence (LCS)**

* Part 1
* Part 2

**24.3 Variation of LCS**

**24.4 Coin Change Count Combinations**

**24.5 Edit Distance Problem**

* DP Solution

**24.6 Longest Increasing Subsequence (LIS)**

* Problem
* O(nlogn) Solution

**24.7 Variation of LIS**

* Part 1
* Part 2

**24.8 Maximum Cuts**

**24.9 Minimum Coins to Make a Value**

**24.10 Minimum Jumps to Reach the End**

**24.11 0-1 Knapsack Problem**

* DP Solution

**Practice Problems**

* Solve various dynamic programming problems.

**25. Advanced Dynamic Programming**

**25.1 Optimal Strategy for a Game**

**25.2 Egg Dropping Puzzle**

* Part 1
* Part 2

**25.3 Count BSTs with n Keys**

**25.4 Maximum Sum with No Two Consecutive**

**25.5 Subset Sum Problem**

* DP Solution

**25.6 Matrix Chain Multiplication**

* DP Solution

**25.7 Palindrome Partitioning**

**25.8 Allocate Minimum Pages**

* Naive Method
* DP Solution