**Week 1**

**1. Array:**

* **Basic Operations:** Insertion, deletion, searching.
* **Simple Manipulations:** Reversal, rotation.
* **Searching Algorithms:** Linear search, binary search.
* **Sorting Algorithms:** Bubble sort, selection sort.
* **Advanced Sorting Techniques:** Merge sort, quick sort.
* **Advanced Operations:** Prefix sums, sliding window technique.
* **Challenges:** Two-pointer technique, matrix manipulation.

**2. String:**

* **Basic Operations:** Concatenation, substring search.
* **Pattern Matching:** Brute force, naive pattern matching.
* **String Reversal and Palindrome Detection:** Basic approaches.
* **Regular Expressions:** Basic syntax, simple pattern matching.
* **String Compression:** Run-Length Encoding, basic compression techniques.
* **Advanced Pattern Matching:** Rabin-Karp algorithm, Boyer-Moore algorithm.
* **Advanced Compression:** Huffman coding, Burrows-Wheeler transform.

**3. Hash Table:**

* **Basic Operations:** Insertion, deletion, searching.
* **Collision Handling:** Chaining, separate chaining.
* **Linear Probing:** Handling collisions using linear probing.
* **Quadratic Probing:** Handling collisions using quadratic probing.
* **Double Hashing:** Resolving collisions with double hashing.
* **Universal Hashing:** Implementing universal hash functions.
* **Perfect Hashing:** Designing perfect hash functions.

**4. Recursion:**

* **Understanding Recursion:** Basic recursive functions (factorial, Fibonacci).
* **Recursive Backtracking:** Simple problems like generating subsets, permutations.
* **Divide and Conquer:** Basic divide-and-conquer problems (binary search).
* **Advanced Backtracking:** Solving complex problems with backtracking (N-Queens problem).
* **Dynamic Programming:** Understanding and implementing dynamic programming solutions.
* **Mutual Recursion:** Problems involving mutual recursion.
* **Tail Recursion Optimization:** Optimizing recursive functions with tail recursion

**Week 2**

**Sorting and Searching:**

* **Sorting Algorithms:** Insertion sort, selection sort.
* **Searching Techniques:** Linear search, basic binary search.
* **Intermediate Sorting Algorithms:** Merge sort, quick sort.
* **Advanced Searching Algorithms:** Interpolation search.
* **Sorting Challenges:** Handling duplicates, stable sorting.
* **Searching Challenges:** Searching in rotated sorted arrays.
* **Specialized Sorting Techniques:** Counting sort, radix sort.

**Matrix:**

* **Matrix Operations:** Addition, subtraction.
* **Basic Matrix Multiplication:** Naive matrix multiplication.
* **Matrix Transpose:** Understanding and implementing matrix transpose.
* **Advanced Matrix Multiplication:** Strassen's algorithm.
* **Determinant and Inverse:** Calculating determinant, finding inverse.
* **Eigenvalues and Eigenvectors:** Basic concepts.
* **Matrix Decomposition:** LU decomposition, QR decomposition.

**Linked List:**

* **Singly Linked List:** Basic operations (insertion, deletion).
* **Doubly Linked List:** Implementing doubly linked list operations.
* **Circular Linked List:** Understanding and implementing circular linked lists.
* **Linked List Reversal:** Reversing a linked list iteratively and recursively.
* **Merging Linked Lists:** Merging two sorted linked lists.
* **Finding Cycle in a Linked List:** Floyd's cycle detection algorithm.
* **Advanced Linked List Operations:** Implementing a skip list.

**Queue:**

* **Queue Operations:** Enqueue, dequeue.
* **Queue Implementation:** Using arrays, linked lists.
* **Circular Queue:** Implementing a circular queue.
* **Priority Queue:** Basic priority queue implementation.
* **Deque:** Implementing double-ended queues.
* **Applications:** BFS traversal, level order traversal.
* **Challenges:** Implementing a stack using queues, design problems.

**Week 3:**

**Tree:**

* **Binary Tree:** Basic properties and operations.
* **Binary Search Tree (BST):** Insertion, deletion, searching.
* **AVL Tree:** Understanding and balancing AVL trees.
* **Binary Heap:** Basic operations (insertion, deletion).
* **Heap Operations:** Heapify, heap sort.
* **Binary Tree Traversal:** Inorder, preorder, postorder traversal.
* **Balanced Binary Trees:** Red-Black trees, B-trees.

**Graph:**

* **Graph Representation:** Adjacency matrix, adjacency list.
* **Graph Traversal:** Depth-first search (DFS), breadth-first search (BFS).
* **Shortest Path Algorithms:** Dijkstra's algorithm, Bellman-Ford algorithm.
* **Minimum Spanning Tree:** Prim's algorithm, Kruskal's algorithm.
* **Topological Sorting:** Understanding and implementing topological sort.
* **Graph Coloring:** Basic graph coloring algorithms.
* **Flow Networks:** Ford-Fulkerson algorithm, Edmonds-Karp algorithm.

**Heap:**

* **Min Heap and Max Heap:** Understanding heap properties.
* **Heap Operations:** Insertion, deletion, heapify.
* **Heap Sort:** Implementing heap sort algorithm.
* **Priority Queue:** Implementing priority queue using heaps.
* **Heap Applications:** kth largest/smallest element, median finding.
* **D-ary Heap:** Understanding and implementing D-ary heaps.
* **Fibonacci Heap:** Understanding advanced heap structures.

**Trie:**

* **Basic Trie:** Implementing basic trie data structure.
* **Trie Operations:** Insertion, deletion, searching.
* **Prefix Tree:** Understanding prefix tree representation.
* **Suffix Tree:** Understanding and implementing suffix tree.
* **Compressed Trie:** Implementing compressed trie for space optimization.
* **Ternary Search Trie:** Understanding and implementing ternary search trie.
* **Applications:** Autocomplete, spell checker.

**Week 4:**

**Interval:**

* **Interval Tree:** Understanding and implementing interval tree.
* **Interval Queries:** Range minimum query, range maximum query.
* **Interval Intersection:** Finding intersections between intervals.
* **Interval Union:** Merging overlapping intervals.
* **Interval Scheduling:** Solving interval scheduling problems.
* **Segment Tree:** Implementing segment tree data structure.
* **Advanced Interval Operations:** Persistent segment trees.

**Dynamic Programming:**

* **Basic Concepts:** Understanding the principles of dynamic programming.
* **Memoization:** Applying memoization to optimize recursive solutions.
* **Tabulation:** Implementing bottom-up dynamic programming solutions.
* **Classic Problems:** Longest common subsequence, knapsack problem.
* **Optimization Problems:** Matrix chain multiplication, rod cutting.
* **Advanced Techniques:** Bitmasking, state compression.
* **Dynamic Programming with Trees and Graphs:** Problems involving trees and graphs.

**Binary:**

* **Binary Representation:** Understanding binary representation of numbers.
* **Bitwise Operations:** Bitwise AND, OR, XOR operations.
* **Bit Manipulation Techniques:** Bit counting, set/reset bits.
* **Bitwise Operators in Algorithms:** Using bitwise operators to solve problems.
* **Applications:** Finding the single unique element, finding subsets.
* **Gray Code:** Understanding and generating Gray code sequences.
* **Advanced Bit Manipulation Techniques:** Next permutation, previous permutation.

**Math:**

* **Number Theory:** Prime numbers, divisibility rules.
* **Modular Arithmetic:** Solving problems using modular arithmetic.
* **Greatest Common Divisor (GCD) and Least Common Multiple (LCM):** Calculating GCD and LCM of numbers.
* **Prime Factorization:** Finding prime factors of a number.
* **Probability Theory:** Basic probability concepts and applications.
* **Combinatorics:** Permutations, combinations.
* **Discrete Mathematics:** Set theory, relations, functions.

**Geometry:**

* **Basic Geometric Shapes:** Points, lines, angles.
* **Coordinate Geometry:** Understanding coordinate systems.
* **Distance and Midpoint:** Calculating distances and midpoints between points.
* **Lines and Slopes:** Equations of lines, slope of a line.
* **Circles:** Equations of circles, tangents, chords.
* **Polygon:** Properties of polygons, convex and concave polygons.
* **Transformations:** Translation, rotation, reflection.