**Data Structures:**

1. **Arrays:**
   * Static arrays
   * Dynamic arrays (like ArrayList in Java, Vector in C++)
2. **Linked Lists:**
   * Singly linked lists
   * Doubly linked lists
   * Circular linked lists
3. **Stacks and Queues:**
   * Basic operations (push, pop, enqueue, dequeue)
   * Implementing with arrays and linked lists
4. **Trees:**
   * Binary Trees
   * Binary Search Trees (BST)
   * AVL Trees (balanced binary search trees)
   * Heap (Min-Heap, Max-Heap)
5. **Graphs:**
   * Representation (Adjacency matrix, Adjacency list)
   * Depth-First Search (DFS)
   * Breadth-First Search (BFS)
6. **Hashing:**
   * Hash tables
   * Hash functions
   * Handling collisions (open addressing, chaining)
7. **Sets and Maps:**
   * Set data structure
   * Map (Dictionary) data structure

**Algorithms:**

1. **Sorting:**
   * Bubble sort
   * Selection sort
   * Insertion sort
   * Merge sort
   * Quick sort
   * Radix sort
2. **Searching:**
   * Linear search
   * Binary search
3. **Recursion:**
   * Understanding recursion
   * Recursive algorithms
4. **Dynamic Programming:**
   * Memoization
   * Bottom-Up approach
   * Examples like Fibonacci, Knapsack problem
5. **Greedy Algorithms:**
   * Basic concepts
   * Examples like Fractional Knapsack, Huffman coding
6. **Graph Algorithms:**
   * Dijkstra's algorithm
   * Bellman-Ford algorithm
   * Kruskal's algorithm
   * Depth-First Search (DFS)
   * Breadth-First Search (BFS)
7. **String Algorithms:**
   * String matching (brute force, KMP algorithm)
   * Longest Common Subsequence (LCS)
   * Edit distance
8. **Sorting and Searching Algorithms for Linked Lists:**
   * Merge Sort for linked lists
   * Binary search on linked lists
9. **Bit Manipulation:**
   * Basics of bitwise operations
   * Bitwise manipulation tricks
10. **Divide and Conquer:**
    * Understanding the paradigm
    * Applying to solve problems

**Other Concepts:**

1. **Big O Notation:**
   * Time and space complexity analysis
2. **Recursion vs Iteration:**
   * Understanding when to use each
3. **Memory (Stack vs Heap):**
   * Understanding memory allocation
4. **Pointers (if using a language that supports them):**
   * Basics of pointer manipulation
5. **Object-Oriented Programming (OOP):**
   * Basic concepts if using an OOP language