**Beginner Level:**

1. Implement Linear Search
2. Implement Binary Search
3. Implement Bubble Sort
4. Implement Selection Sort
5. Implement Insertion Sort
6. Find the factorial of a number (recursive and iterative)
7. Generate the nth Fibonacci number (recursive and iterative)
8. Find the greatest common divisor (GCD) of two numbers (Euclidean algorithm)
9. Find the least common multiple (LCM) of two numbers
10. Check if a number is prime
11. Generate all prime numbers up to n (Sieve of Eratosthenes)
12. Reverse a string
13. Check if a string is a palindrome
14. Find the maximum and minimum element in an array
15. Compute the power of a number (exponentiation by squaring)
16. Merge two sorted arrays
17. Find the first non-repeating character in a string
18. Implement Depth First Search (DFS) for a graph
19. Implement Breadth First Search (BFS) for a graph
20. Find the length of the longest common subsequence (LCS)

**Medium Level:**

1. Implement Merge Sort
2. Implement Quick Sort
3. Find the kth smallest/largest element in an array
4. Find the longest increasing subsequence in an array
5. Find the shortest path in a graph (Dijkstra's algorithm)
6. Implement a stack using arrays
7. Implement a queue using arrays
8. Implement a binary search tree (BST)
9. Implement a heap and perform heap operations
10. Check if two strings are anagrams
11. Implement the Knuth-Morris-Pratt (KMP) pattern matching algorithm
12. Implement the Rabin-Karp string matching algorithm
13. Solve the 0/1 Knapsack problem using dynamic programming
14. Solve the Coin Change problem using dynamic programming
15. Find the maximum sum subarray (Kadane's algorithm)
16. Implement a hash table
17. Find the lowest common ancestor (LCA) in a binary tree
18. Check if a binary tree is balanced
19. Flatten a binary tree to a linked list
20. Implement the Union-Find algorithm with path compression

**Advanced Level:**

1. Implement the Floyd-Warshall algorithm for all-pairs shortest paths
2. Implement the Bellman-Ford algorithm for single-source shortest path
3. Implement the A\* search algorithm
4. Find the strongly connected components in a graph (Kosaraju's algorithm)
5. Find the articulation points in a graph
6. Find the bridges in a graph
7. Implement a trie (prefix tree)
8. Implement a suffix tree
9. Solve the Traveling Salesman Problem (TSP) using dynamic programming
10. Implement the Boyer-Moore string matching algorithm