**Month 1: Fundamentals and Arrays**

**Topics:**

* Python Basics Review (if required): Loops, conditionals, functions, classes, and file handling.
* Big-O Notation: Understand time and space complexity.
* Arrays/Lists:
  + Traversals, insertions, deletions, and basic operations.
  + Prefix sums, sliding window technique, two-pointer technique.

**Problem Practice:**

* Easy problems:
  + Reverse an array.
  + Find the largest/smallest element.
  + Remove duplicates from a sorted array.
* Medium problems:
  + Two Sum.
  + Subarray with given sum.
  + Rotate an array.

**Resources:**

* **Book:** *Introduction to Algorithms* (Cormen) - Focus on the introduction and arrays chapter.
* **Platform:** LeetCode Arrays Problems.

**Month 2: Recursion, Strings, and Linked Lists**

**Topics:**

1. **Recursion**:
   * Base cases, recursion tree, and backtracking.
   * Problems: Fibonacci, Tower of Hanoi.
2. **Strings**:
   * Manipulations, substrings, pattern matching (basic).
   * Palindromes, anagrams, and reverse words in a string.
3. **Linked Lists**:
   * Singly linked list: traversal, insertion, deletion.
   * Reverse a linked list, detect a cycle.

**Problem Practice:**

* Easy:
  + Factorial using recursion.
  + Reverse a string.
  + Detect a loop in a linked list.
* Medium:
  + Longest substring without repeating characters.
  + Merge two sorted linked lists.
  + Reverse a linked list in groups.

**Resources:**

* **Videos:** Abdul Bari on Recursion and Linked Lists.
* **Platform:** Solve problems on GeeksforGeeks or LeetCode.

**Month 3: Stacks, Queues, and Hashing**

**Topics:**

1. **Stacks and Queues**:
   * Stack: Implement using lists, balanced parentheses, Next Greater Element.
   * Queue: Circular queue, deque.
2. **Hashing (Dictionaries and Sets)**:
   * Hash tables, hash sets.
   * Applications in frequency counting and duplicate detection.

**Problem Practice:**

* Easy:
  + Valid Parentheses.
  + Implement stack/queue using arrays.
* Medium:
  + LRU Cache (Least Recently Used Cache).
  + Evaluate Reverse Polish Notation.
  + Longest Consecutive Sequence.

**Resources:**

* **Platform:** Practice Stack Problems on LeetCode.

**Month 4: Trees and Graphs**

**Topics:**

1. **Trees**:
   * Binary Trees: Traversals (Inorder, Preorder, Postorder), height, diameter.
   * Binary Search Trees (BST): Insert, delete, search.
2. **Graphs**:
   * Representations: Adjacency matrix/list.
   * Traversals: BFS, DFS.
   * Graph algorithms: Dijkstra's, Kruskal's, Prim's.

**Problem Practice:**

* Easy:
  + Level order traversal.
  + Symmetric tree.
* Medium:
  + Lowest Common Ancestor.
  + Shortest path in an unweighted graph.

**Resources:**

* **Books:** *Data Structures and Algorithm Analysis in Python* by Mark Allen Weiss.
* **Platform:** Solve Tree and Graph problems on HackerRank.

**Month 5: Greedy, Divide & Conquer, and Dynamic Programming**

**Topics:**

1. **Greedy Algorithms**:
   * Coin change problem, activity selection.
   * Huffman encoding.
2. **Divide & Conquer**:
   * Merge Sort, Quick Sort.
   * Binary search variations.
3. **Dynamic Programming**:
   * Memoization and tabulation.
   * Problems: Longest Common Subsequence, Knapsack Problem, Fibonacci.

**Problem Practice:**

* Easy:
  + Climbing Stairs (DP).
  + Coin Change (Greedy).
* Medium:
  + Longest Palindromic Subsequence.
  + Partition Equal Subset Sum.

**Resources:**

* **Platform:** Dynamic Programming Problems on LeetCode.

**Month 6: Advanced Algorithms and Mock Interviews**

**Topics:**

1. **Advanced Algorithms**:
   * Trie (Prefix Trees): Auto-suggestions, word search.
   * Segment Trees: Range queries.
   * Union-Find: Connected components, cycle detection.
2. **Problem Solving Practice**:
   * Focus on medium and hard-level problems.
   * Solve problems with varying patterns to build confidence.

**Mock Interviews:**

* Schedule mock interviews on platforms like Pramp or InterviewBit.
* Practice coding under time constraints.

**Resources:**

* **Platform:** Explore real interview problems on LeetCode.

**Tools to Track Progress**

1. **Daily Problem Solving:**
   * Solve 2–3 problems per day from the topic you're studying.
2. **Review and Reflect:**
   * Revisit problems you couldn’t solve after a week.
3. **Competitive Coding Practice:**
   * Participate in contests on Codeforces or CodeChef.

**Results by End of 6 Months:**

1. Strong foundation in DSA.
2. Confidence to solve medium to hard-level problems.
3. Preparedness for technical interviews.

Would you like a week-by-week breakdown or specific problem recommendations? 😊

4o

**You said:**

can you gave me problem statement

**ChatGPT said:**

ChatGPT

Here’s a **list of DSA problem statements** aligned with the 6-month plan. These are categorized by topic and difficulty for progressive learning.

**Month 1: Arrays and Lists**

**Easy:**

1. **Find the Maximum and Minimum in an Array**
   * Input: arr = [3, 5, 1, 2, 4]
   * Output: Maximum = 5, Minimum = 1.
2. **Reverse an Array**
   * Input: arr = [1, 2, 3, 4, 5]
   * Output: [5, 4, 3, 2, 1].
3. **Find All Unique Elements**
   * Input: arr = [1, 2, 2, 3, 4, 4, 5]
   * Output: [1, 3, 5].

**Medium:**

1. **Two Sum**
   * Given an array and a target, find two numbers such that their sum equals the target.
   * Input: nums = [2, 7, 11, 15], target = 9
   * Output: [0, 1].
2. **Find Subarray with Given Sum**
   * Input: arr = [1, 4, 20, 3, 10, 5], sum = 33
   * Output: Subarray [20, 3, 10].

**Month 2: Recursion, Strings, and Linked Lists**

**Recursion:**

1. **Factorial of a Number**
   * Input: n = 5
   * Output: 120.
2. **Generate All Subsequences of a String**
   * Input: "abc"
   * Output: ["", "a", "b", "c", "ab", "ac", "bc", "abc"].

**Strings:**

1. **Check If String is a Palindrome**
   * Input: "madam"
   * Output: True.
2. **Group Anagrams**
   * Input: ["eat", "tea", "tan", "ate", "nat", "bat"]
   * Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]].

**Linked Lists:**

1. **Reverse a Linked List**
   * Input: 1 → 2 → 3 → 4 → 5
   * Output: 5 → 4 → 3 → 2 → 1.
2. **Detect Cycle in a Linked List**
   * Input: 1 → 2 → 3 → 4 → 2 (cycle)
   * Output: True.

**Month 3: Stacks, Queues, and Hashing**

**Stacks and Queues:**

1. **Validate Parentheses**
   * Input: "({[]})"
   * Output: True.
2. **Implement a Min Stack**
   * Support push(x), pop(), top(), and getMin() operations.

**Hashing:**

1. **Find the First Non-Repeating Character in a String**
   * Input: "leetcode"
   * Output: "l".
2. **Longest Substring Without Repeating Characters**
   * Input: "abcabcbb"
   * Output: 3 (abc).

**Month 4: Trees and Graphs**

**Trees:**

1. **Level Order Traversal of a Binary Tree**
   * Input: [3, 9, 20, None, None, 15, 7]
   * Output: [[3], [9, 20], [15, 7]].
2. **Lowest Common Ancestor in a Binary Search Tree**
   * Input: root = [6, 2, 8, 0, 4, 7, 9], p = 2, q = 8
   * Output: 6.

**Graphs:**

1. **Find Connected Components in an Undirected Graph**
   * Input: Edges = [[0, 1], [1, 2], [3, 4]]
   * Output: 2.
2. **Shortest Path in a Weighted Graph (Dijkstra’s Algorithm)**
   * Input: Graph = {0: [(1, 4), (2, 1)], 1: [(2, 2), (3, 1)], 2: [(3, 5)]}
   * Output: Shortest distance from node 0.

**Month 5: Greedy, Divide & Conquer, and Dynamic Programming**

**Greedy:**

1. **Activity Selection Problem**
   * Input: Start = [1, 3, 0, 5, 8, 5], End = [2, 4, 6, 7, 9, 9]
   * Output: 4 activities.
2. **Huffman Encoding**
   * Encode a given string using Huffman encoding.

**Divide & Conquer:**

1. **Merge Sort Implementation**
   * Input: [3, 1, 4, 1, 5, 9, 2]
   * Output: [1, 1, 2, 3, 4, 5, 9].
2. **Search in Rotated Sorted Array**
   * Input: nums = [4,5,6,7,0,1,2], target = 0
   * Output: 4.

**Dynamic Programming:**

1. **Knapsack Problem (0/1)**
   * Input: weights = [1, 2, 3], values = [6, 10, 12], capacity = 5
   * Output: 22.
2. **Longest Common Subsequence**
   * Input: X = "ABCBDAB", Y = "BDCABC"
   * Output: 4 (BDAB).

**Month 6: Advanced Algorithms**

**Advanced Algorithms:**

1. **Trie Implementation**
   * Insert and search words in a dictionary using a trie.
2. **Find Bridges in a Graph**
   * Input: Graph = [[0, 1], [1, 2], [2, 0], [1, 3]]
   * Output: Bridge: [1, 3].

**Mock Problems for Practice:**

1. **Find Median from Data Stream**
   * Input: Stream [2, 3, 4]
   * Output: Median = 3.
2. **N-Queens Problem**
   * Input: N = 4
   * Output: All valid arrangements.