AlgoBoost DSA Roadmap 🚀

Month 1: C++ Basics and Problem-Solving Foundation

Week 1-2: Master C++ Basics 📚

1. **Input/Output**
    - cin, cout, getline, printf, scanf.
2. **Data Types, Operators, and Variables**
3. **Control Structures**
    - If-else, switch-case, loops (for, while, do-while).
4. **Functions**
    - Pass-by-value, pass-by-reference, inline functions.
5. **Arrays and Strings**
    - One-dimensional and two-dimensional arrays.
    - String operations using the string library.

Resources:

- Practice: Basic problems on CodeChef, HackerRank, Leetcode, HackerEarth, CodeForces.

Week 3-4: Pointers, STL, and Basic Problem-Solving 🔧

1. **Pointers and Memory Management**
    - Basics of pointers, dynamic memory allocation (new, delete).
2. **Object-Oriented Programming (OOP)**
    - Classes, objects, constructors, destructors, inheritance, polymorphism.
3. **Standard Template Library (STL)**
    - Vectors, Maps, Sets, and Stacks.
    - Understand iterators and basic STL functions.
4. **Problem-Solving**
    - Solve 50 easy problems on arrays, strings, and STL.

Resources:

- STL: Watch videos from Striver or Love Babbar
- Practice: Start on LeetCode or GFG.

Month 2: Recursion and Basic Data Structures

Week 1-2: Recursion and Backtracking 🔄

1. Understand the concept of recursion and base case.
2. Solve problems:
    - Factorial, Fibonacci, GCD, power of a number.
    - Tower of Hanoi, subsets, permutations.

Week 3-4: Basic Data Structures 🛠️

1. **Linked List**
    - Singly Linked List: Insertion, deletion, traversal.
    - Doubly Linked List.
2. **Stacks and Queues**
    - Implement using arrays and linked lists.
    - Use STL stacks/queues for problem-solving.

Practice Problems:

- GFG Linked List problems.
- Striver's Video Lectures.
- LeetCode problems on Recursion, Linked List, Stacks and Queues.

Month 3: Advanced Data Structures

Week 1-2: Trees 🌳

1. **Binary Trees**
    - Preorder, Inorder, Postorder traversals (recursive + iterative).
    - Level-order traversal.
2. **Binary Search Tree (BST)**
    - Insert, delete, search.

Week 3-4: Heaps and Hashing ⛰️

1. **Heaps**
    - Min-heap and max-heap (implement using arrays).
    - Priority queues using STL.
2. **Hashing**
    - Hash maps (unordered_map in STL).
    - Collision handling using chaining.

Practice Problems:

- Solve 30 tree problems (LeetCode, GFG).
- Aditya Verma for Heap.
- Focus on understanding concepts deeply.

Month 4: Graphs

Week 1-2: Graph Basics 🌐

1. **Representation**
    - Adjacency matrix, adjacency list.
2. **Traversals**
    - BFS (Breadth-First Search).
    - DFS (Depth-First Search).

Week 3-4: Backtracking □

1. N-Queens Problem.
2. Rat in a Maze.
3. Sudoku Solver.

Practice:

- Watch Videos of Striver on Graph
- Solve graph problems on GFG or LeetCode.
- Use SDE Sheet for problem lists.

Month 5: Dynamic Programming (DP) 💡

Week 1-2: Introduction to DP

1. Understand memoization and tabulation.
2. Solve classical problems:
    - Fibonacci, Climbing Stairs, Coin Change, Knapsack Problem.

Week 3-4: Advanced DP

1. Longest Common Subsequence (LCS).
2. Longest Increasing Subsequence (LIS).
3. Matrix Chain Multiplication.

Practice:

- Solve 30 DP problems.
- Striver, Aditya Verma, or LoveBabbar.
- Use LeetCode, GFG, or Striver's DP Series.

Month 6: Advanced Topics and Competitive Programming

Week 1-2: Greedy Algorithms and Bit Manipulation □

1. **Greedy Algorithms**
    - Activity selection, Huffman coding.
2. **Bit Manipulation**
    - Basic operations, subsets, toggling bits.

Week 3-4: Competitive Programming Practice 🏆

1. Practice problems on:
    - Codeforces Div 2 contests.
    - LeetCode Weekly contests.
2. Revise all topics.
3. Solve 100 mixed problems (arrays, graphs, DP, trees).

    Resources:

    Watch Striver Videos on these topics.

General Tips 💡

1. **Consistency is key** – Aim for 2-3 hours daily.
2. Track progress using sheets like ALgoBoost DSA Sheet or Striver's SDE Sheet or Love Babbar's 450 DSA Questions.
3. Learn and code simultaneously; do not just watch tutorials.
4. Use debuggers and tools like GDB to understand code execution.
5. Participate in contests to improve speed and accuracy.

By the end of 6 months, you will have a solid foundation in C++ and DSA, and you'll be well-prepared for coding interviews or competitive programming contests! 🎯
