Detailed two-month plan for covering DSA (Data Structures and Algorithms) with C++:

Week 1: Basics and Foundations

Day 1-2:

- Set up your environment: Install an IDE (e.g., VS Code, Code::Blocks).
- Revise C++ basics:
    - Input/output, loops, conditionals.
    - Functions, arrays, and strings.

Day 3-5:

- Dive into time and space complexity (Big O, Ω, Θ).
- Learn recursion basics and solve simple problems.

Day 6-7:

- Explore sorting and searching algorithms:
    - Bubble, Selection, Insertion Sort, Binary Search.
    - Solve basic problems on arrays and sorting.

Week 2: Arrays, Strings, and Pointers

Day 8-10:

- Advanced array problems:
    - Sliding window, prefix sums, Kadane's algorithm.
- Solve problems on platforms like LeetCode/Codeforces.

Day 11-12:

- Strings:
    - Pattern matching (KMP, Rabin-Karp).
    - String manipulation problems.

Day 13-14:

- Pointers and dynamic memory allocation in C++.
- Practice dynamic arrays and matrices.

Week 3: Linked Lists and Stacks

Day 15-17:

- Learn singly and doubly linked lists:
    - Basic operations (insert, delete, reverse).
    - Solve problems like detecting cycles.

Day 18-20:

- Stacks and their applications:
    - Implement using arrays and linked lists.
    - Solve problems (e.g., balanced parentheses, next greater element).

Week 4: Queues, Deques, and Hashing

Day 21-23:

- Queues and priority queues:
    - Implement circular queues and dequeue.
    - Problems on BFS and sliding window maximum.

Day 24-25:

- Hashing:
    - Learn hash maps, collision handling.
    - Solve problems like 2-sum, frequency counters.

Day 26-28:

- Revise and solve mixed problems from arrays, strings, and stacks.

Week 5: Trees

Day 29-31:

- Learn binary trees:
    - Basic traversals (Inorder, Preorder, Postorder).
    - Solve problems on tree traversal.

Day 32-34:

- Binary Search Tree (BST):
    - Insert, delete, and search operations.
    - Problems like finding LCA, floor, and ceil.

Day 35:

- Revision and problem-solving on trees.

Week 6: Graphs

Day 36-38:

- Graph representation (Adjacency Matrix, List).
- Learn DFS and BFS traversals.

Day 39-40:

- Topological sorting, cycle detection.
- Solve problems on undirected and directed graphs.

Day 41-42:

- Shortest path algorithms:
    - Dijkstra's and Floyd-Warshall.

Week 7: Dynamic Programming (DP)

Day 43-46:

- Learn DP basics:
    - Memoization vs Tabulation.
    - Solve classic problems like Fibonacci, Knapsack.

Day 47-48:

- Advanced DP problems:
    - Longest Increasing Subsequence, Matrix Chain Multiplication.

Week 8: Advanced Topics and Mock Interviews

Day 49-51:

- Advanced topics:
    - Tries, Segment Trees, Fenwick Tree.
    - Solve problems using these data structures.

Day 52-53:

- Backtracking:
    - N-Queens, Sudoku Solver.
    - Problems on permutations/combinations.

Day 54-56:

- Revision:
    - Solve mixed problems from previous weeks.

Day 57-58:

- Mock interviews:
    - Simulate coding interviews on platforms like LeetCode or HackerRank.

Day 59-60:

- Revise and solidify weak areas.

Tools and Platforms:

- **Platforms for practice:** LeetCode, Codeforces, GeeksforGeeks, HackerRank.
- **Books:** "Introduction to Algorithms" by Cormen (CLRS), or "Data Structures and Algorithms Made Easy" by Narasimha Karumanchi.

How does this look? Would you like me to modify or expand on anything?