Guide2Code - Data Structures & Algorithms (DSA) Roadmap

🟢 Phase 1: Beginner Level

📚 Topics to Learn:

1. Introduction to DSA (Why DSA?, Complexity Analysis - Big O Notation)
2. **Arrays** (1D & 2D Arrays, Operations: Insert, Delete, Search)
3. **Strings** (String Manipulation, Palindromes, Anagrams)
4. **Linked Lists** (Singly, Doubly & Circular Linked Lists)
5. **Stacks & Queues** (Stack using Arrays & Linked Lists, Queue Variants - Circular, Priority, Deque)
6. **Recursion** (Understanding Recursion, Tail Recursion, Backtracking Basics)
7. **Sorting Algorithms** (Bubble, Selection, Insertion, Merge, Quick Sort)
8. **Searching Algorithms** (Linear Search, Binary Search & Variants)
9. **Hashing** (Hash Functions, Collision Handling - Chaining, Open Addressing)
10. **Basic Mathematical Algorithms** (GCD, LCM, Prime Numbers, Fibonacci, Factorial)

🛠 Beginner Project Ideas:

- **Sorting Visualizer** – Graphical representation of sorting algorithms
- **Palindrome Checker** – Check if a word/sentence is a palindrome
- **Basic Calculator** – Implement operations using stacks
- **Anagram Finder** – Check if two words are anagrams
- **Simple Contact Book** – Store and search contacts using arrays

---

🟡 Phase 2: Intermediate Level

📚 Topics to Learn:

1. **Recursion & Backtracking** (Sudoku Solver, N-Queens, Rat in a Maze)
2. **Advanced Sorting Algorithms** (Heap Sort, Bucket Sort, Radix Sort)
3. **Linked List Operations** (Reversing a List, Detecting Loops, Merging)
4. **Stack & Queue Applications** (Infix to Postfix Conversion, Expression Evaluation)

5. **Binary Trees & BST** (Tree Traversals, Height of a Tree, Lowest Common Ancestor)
6. **Heap Data Structure** (Min Heap, Max Heap, Heap Sort, Priority Queue)
7. **Graph Theory** (Graph Representation, DFS, BFS, Shortest Paths - Dijkstra, Floyd Warshall)
8. **Greedy Algorithms** (Huffman Encoding, Activity Selection, Kruskal's & Prim's Algorithm)
9. **Dynamic Programming (DP)** (Fibonacci, Knapsack, Longest Common Subsequence)
10. **Trie Data Structure** (Prefix Trees, Auto-Completion, Searching in Tries)

🛠 **Intermediate Project Ideas:**

- **Knight's Tour Problem** – Solve using backtracking
- **Expression Evaluator** – Convert & evaluate mathematical expressions
- **Dictionary Auto-Suggester** – Implement auto-suggestions using Tries
- **Maze Solver** – Solve a maze using DFS or BFS
- **Text Compression Tool** – Implement Huffman Encoding

🔴 Phase 3: Advanced Level

📚 **Topics to Learn:**

1. **Segment Trees & Fenwick Trees** (Range Queries, Lazy Propagation)
2. **Graph Algorithms** (Bellman-Ford, Topological Sorting, Strongly Connected Components)
3. **Advanced Dynamic Programming** (Subset Sum, Matrix Chain Multiplication, Edit Distance)
4. **Bit Manipulation Techniques** (Bitwise Operations, XOR Tricks, Subsets)
5. **Disjoint Set Union (DSU)** (Union-Find, Path Compression, Kruskal's Algorithm)
6. **String Algorithms** (KMP, Rabin-Karp, Z Algorithm, Suffix Trees & Arrays)
7. **Network Flow & Matching** (Ford-Fulkerson Algorithm, Bipartite Matching)
8. **Game Theory Algorithms** (Minimax, Alpha-Beta Pruning)
9. **Approximation & Randomized Algorithms** (Monte Carlo, Las Vegas Algorithms)

10. **Competitive Programming Techniques** (Efficient Code Writing, Optimized Approach Selection)

🛠 **Advanced Project Ideas:**

- **AI-Based Tic-Tac-Toe** – Implement Minimax Algorithm
- **Shortest Path Visualizer** – Show Dijkstra & A* Algorithm working
- **Real-Time Spell Checker** – Implement using Trie & Edit Distance
- **Stock Market Predictor** – Use DP & ML for stock trend analysis
- **Online Pathfinding System** – Implement A* or Dijkstra for real-world navigation