**🛣️ DSA Learning Pathway**

**1. Foundations (must be solid first)**

* Time Complexity (Big-O, Big-Θ, Big-Ω)
* Space Complexity
* Recursion basics (factorial, Fibonacci, backtracking intro)

**2. Core Data Structures**

* **Arrays & Strings**
  + Traversals, prefix sums, sliding window, two pointers
  + Subarray/subsequence problems
* **Hashing**
  + HashSet (uniqueness, duplicates)
  + HashMap (frequency counts, lookup)
* **Linked List**
  + Singly, doubly, fast & slow pointers
  + Reversal, cycle detection (Floyd’s algorithm)

**3. Stack & Queue**

* Stack (next greater element, parentheses problems, min stack)
* Queue (basic operations, sliding window max with deque)
* Monotonic Stack/Queue (for optimization problems)

**4. Recursion & Backtracking**

* Subsets / Combinations / Permutations
* N-Queens, Rat in a Maze, Sudoku Solver
* Recursive Tree traversals

**5. Trees**

* Binary Tree traversals (DFS, BFS, preorder, inorder, postorder)
* Binary Search Tree (search, insert, delete, kth smallest/largest)
* Lowest Common Ancestor (LCA)
* Trie (prefix problems, autocomplete)

**6. Heaps & Priority Queue**

* Min Heap, Max Heap (k largest/smallest problems)
* Heap sort
* Priority-based scheduling problems

**7. Graphs**

* Representations: adjacency list, adjacency matrix
* Traversals: DFS, BFS
* Shortest Path: Dijkstra, Bellman-Ford
* MST: Kruskal, Prim
* Topological Sort
* Union-Find (Disjoint Set Union)

**8. Advanced Topics**

* **Dynamic Programming (DP)**
  + 1D DP (Fibonacci, climbing stairs, coin change)
  + 2D DP (grid paths, longest common subsequence, knapsack)
  + DP on subsequences, DP on strings, DP with bitmasks
* **Greedy Algorithms** (interval scheduling, Huffman coding)
* **Binary Search on Answer** (searching in solution space)
* **Segment Trees / Fenwick Tree (BIT)** (range queries, prefix sums)

**9. Special Topics (when aiming FAANG-level)**

* Bit Manipulation (XOR problems, subsets)
* Advanced Graphs (Tarjan’s, Kosaraju, bipartite check)
* Advanced DP (digit DP, DP + bitmask, DP on trees)
* String Algorithms (KMP, Rabin-Karp, Z-function, suffix arrays)

**10. Contest/Interview Prep**

* Mix problems across topics
* Focus on **patterns**:
  + Sliding Window
  + Two Pointers
  + Binary Search
  + Prefix/Suffix tricks
  + Recursion → DP transitions
* Timed practice (LeetCode, Codeforces, AtCoder)

⚡ **Rule of Thumb**:

* If problem involves *unique elements / duplicates* → HashSet.
* If problem involves *counts / mapping* → HashMap.
* If problem involves *sequence* → Array / LinkedList.
* If problem involves *parentheses / order of ops* → Stack.
* If problem involves *shortest path / connectivity* → Graph.
* If problem involves *optimization with choices* → DP or Greedy.

**🛣️ DSA Concepts Roadmap**

**1. Basics**

* Time Complexity
* Space Complexity
* Recursion

**2. Core Data Structures**

* Arrays
* Strings
* Hashing (HashSet, HashMap)
* Linked List

**3. Stack & Queue**

* Stack
* Queue
* Deque
* Monotonic Stack/Queue

**4. Recursion & Backtracking**

* Subsets
* Permutations
* Combinations
* Classic backtracking problems (N-Queens, Sudoku, Maze)

**5. Trees**

* Binary Trees
* Binary Search Trees (BST)
* Traversals (DFS, BFS, preorder, inorder, postorder)
* Lowest Common Ancestor (LCA)
* Trie

**6. Heaps & Priority Queue**

* Min Heap
* Max Heap
* Heap Sort
* Priority Queue

**7. Graphs**

* Representation (adjacency list / matrix)
* DFS
* BFS
* Shortest Path (Dijkstra, Bellman-Ford)
* Minimum Spanning Tree (Kruskal, Prim)
* Topological Sort
* Union-Find (Disjoint Set Union)

**8. Advanced Algorithms**

* Dynamic Programming (DP)
* Greedy Algorithms
* Binary Search on Answer
* Segment Trees
* Fenwick Tree (BIT)

**9. Special Topics**

* Bit Manipulation
* Advanced Graphs (Tarjan’s, Kosaraju, Bipartite check)
* Advanced DP (digit DP, bitmask DP, DP on trees)
* String Algorithms (KMP, Rabin-Karp, Z-function, suffix arrays/tries)

**10. Contest / Interview Patterns**

* Sliding Window
* Two Pointers
* Prefix/Suffix Tricks
* Recursion → DP transitions