NS LONI 2k25-26

Level 1: Basics (SDE-1 / Entry-Level)

Goal: Master core concepts + solve **100+ easy/medium problems**.

1. Arrays & Strings

- **Key Skills**: Sliding Window (e.g., longest substring), Two Pointers (e.g., palindrome checks), Prefix Sum (e.g., subarray sums).
- **Critical Problems**:
    - Two Sum (HashMap pattern)
    - Maximum Subarray (Kadane's Algorithm)

2. Linked Lists

- **Key Skills**: Cycle detection (Floyd's Algorithm), dummy nodes for edge cases.
- **Critical Problems**:
    - Reverse Linked List (Iterative + Recursive)

3. Stacks & Queues

- **Key Skills**: Use stacks for LIFO operations (e.g., parsing expressions), queues for BFS.
- **Critical Problems**:
    - Valid Parentheses (Classic stack use-case)

4. Sorting & Searching

- **Key Skills**: Master Binary Search variations (rotated arrays, duplicates).
- **Critical Problems**:
    - Find First/Last Position in Sorted Array

5. Recursion & Backtracking

- **Key Skills**: Base cases + decision trees (e.g., permutations).
- **Critical Problems**:
    - N-Queens (Backtracking template)

1. Pattern Printing	5. Searching	9. Stack
2. Array	6. Recursion	10. Queue
3. String	7. Backtracking	11. Linked List
4. Sorting	8. Matrix	

Level 2: Intermediate (SDE-2 / Mid-Level)

Goal: Solve **200+ medium/hard problems** + optimize time/space complexity.

1. Trees & Binary Trees

- **Key Skills**: Inorder traversal for BST validation, LCA using DFS/BFS.
- **Critical Problems**:
    - Validate BST (Inorder traversal trick)

2. Heaps & Priority Queues

- **Key Skills**: Min/Max heaps for Kth-largest/smallest elements.
- **Critical Problems**:
    - Merge K Sorted Lists (Heap-based merging)

3. Hashing

- **Key Skills**: Subarray sum tricks (prefixSum + HashMap).
- **Critical Problems**:
    - Subarray Sum Equals K

4. Dynamic Programming

- **Key Skills**: Memoization vs tabulation; start with 1D DP (climbing stairs) → 2D DP (LCS).
- **Critical Problems**:
    - Longest Increasing Subsequence (Binary search optimization)

5. Graphs

- **Key Skills**: BFS for shortest unweighted paths; DFS for connected components.
- **Critical Problems**:
    - Course Schedule (Topological sorting)

1. Trees

2. Binary Tree

3. AVL Trees

4. Binary Search Tree

5. Sliding Window

6. Heaps

7. Priority Queues

8. Hashing

9. Dynamic Programming

10. Graphs

Level 3: Advanced (SDE-3 / Senior-Level)

Goal: Solve **300+ hard problems** + integrate system design principles (e.g., scalable Trie storage).

1. Advanced Trees

- **Key Skills**: Segment Trees for range queries; AVL/RB Trees for self-balancing BSTs.
- **Critical Problems**:
    - Range Sum Query Mutable

2. Advanced Graphs

- **Key Skills**: Dijkstra's for weighted graphs; Union-Find for MSTs (Kruskal's Algorithm).

3. Advanced DP & Greedy

- **Key Skills**: Knapsack variations → space optimization; greedy intervals (Merge Intervals).

4. Tries & Bit Manipulation

- **Key Skills**: XOR tricks (Single Number), Trie-based autocomplete systems.

1. Advanced Trees
2. Advanced Graph
3. Advanced DP
4. Tries
5. Bit Manipulation
6. Greedy
7. More. Follow TUF

--------------------- **Hey, NS LONI**