🔧 **Core Data Structures**

You should master the operations (insert, delete, search, etc.), use-cases, and time complexities for each.

1. **Arrays**
   * Two-pointer problems
   * Sliding window
   * Kadane’s algorithm (max subarray)
   * Merge intervals
2. **Strings**
   * Palindrome checks
   * Anagram grouping
   * Substring search (KMP, Rabin-Karp)
   * Longest common subsequence
3. **Linked Lists**
   * Reverse a linked list
   * Detect cycle (Floyd’s Algorithm)
   * Merge two sorted lists
   * LRU Cache
4. **Stacks & Queues**
   * Min stack
   * Evaluate Reverse Polish Notation
   * Queue using stacks
   * Sliding window maximum
5. **Hashing / Hash Tables**
   * Two sum / Four sum
   * Longest substring without repeating chars
   * Top K frequent elements
6. **Trees**
   * Binary Tree / BST traversals (DFS, BFS)
   * Lowest Common Ancestor
   * Balanced tree check
   * Serialize/Deserialize tree
7. **Tries**
   * Word search / prefix trees
   * Autocomplete systems
8. **Graphs**
   * DFS / BFS
   * Topological Sort
   * Dijkstra’s Algorithm
   * Union-Find (Cycle detection, connected components)
9. **Heaps / Priority Queues**
   * Median in a data stream
   * K closest points
   * Merge k sorted lists
10. **Intervals**
    * Merge intervals
    * Meeting rooms
    * Insert and delete interval

📈 **Algorithms**

These are vital for solving more complex problems.

1. **Sorting**
   * Quick sort, Merge sort
   * Counting sort / Bucket sort
   * Custom comparator sorting (e.g., for strings)
2. **Searching**
   * Binary search (classic and on rotated arrays)
   * Search in 2D matrix
3. **Recursion / Backtracking**
   * Sudoku solver
   * N-Queens
   * Permutations and combinations
   * Subsets
4. **Dynamic Programming**
   * 0/1 Knapsack
   * Longest increasing subsequence
   * Edit distance
   * DP on trees / DP on strings / DP with memoization
5. **Greedy Algorithms**
   * Activity selection
   * Huffman encoding
   * Interval scheduling
6. **Bit Manipulation**
   * Single number
   * Count set bits
   * XOR problems
7. **Math / Combinatorics**
   * GCD/LCM
   * Sieve of Eratosthenes
   * Modular arithmetic
   * Combinations / permutations with constraints

🌐 **Frontend Engineer-Specific Additions**

While most DSA is the same, frontend engineers should be familiar with:

* DOM tree traversal algorithms
* Event loop, async/await mechanics (JavaScript)
* String parsing (e.g., HTML/XML or custom tags)
* Mini-parser or expression evaluator
* Debounce/throttle logic simulation
* Caching strategies (LRU/FIFO cache)
* Data visualization algorithms (like layouting trees or graphs)