Zero-to-Hero DSA Crash Course.

Topic 1: Complexity (The Speed of Code)

They usually ask: "What is the Time Complexity of X?"

Concept: Big O Notation (O) measures how "slow" an algorithm gets as data grows.

- **$O(1)$ (Constant):** Instant. (e.g., Accessing an array index `arr[5]`).
- **$O(\log n)$ (Logarithmic):** Very fast. Cuts data in half every time. (e.g., **Binary Search**).
- **$O(n)$ (Linear):** Reading every item once. (e.g., Finding an item in an unsorted list).
- **$O(n^2)$ (Quadratic):** Slow. Nested loops. (e.g., **Bubble Sort**).

Topic 2: Linked Lists (The Chain)

Concept: Unlike Arrays (continuous block), a Linked List is a chain of nodes scattered in memory.

- **Node:** Contains two things: **Data** + **Pointer** (Address of next node).
- **Head:** The start of the list.
- **Null:** The end of the list.

Key Exam Trick (Visualizing Pointers):

If you see code like p->next = q->next, draw it.

- `p->next` means "The arrow coming out of P".
- If the arrow moves, the link breaks.
- **Insertion:** To add a node in the middle, you must attach the *new node* to the *next node* **first**, then attach the *previous node* to the *new node*. (Otherwise, you drop the chain).

Topic 3: Stacks & Queues (The Rules)

These are guaranteed 1-2 questions.

A. Stack (LIFO - Last In First Out)

- **Analogy:** A stack of plates. You put the last one on top, and you must take that one off first.
- **Operations:**
  - **Push:** Add to top.
  - **Pop:** Remove from top.
- **Exam Question:** "Which data structure is used for **Recursion** or **Undo** buttons?" $\to$ **Stack**.

B. Queue (FIFO - First In First Out)

- **Analogy:** A line for movie tickets. The person who comes first gets served first.
- **Operations:**
    - **Enqueue:** Add to rear (back).
    - **Dequeue:** Remove from front.
- **Exam Question:** "Which data structure is used for **Printer Spooling** or **BFS**?" $\to$ **Queue**.

C. Postfix Evaluation (The Calculation Question)

You will get an expression like: 5 3 + 2 *

How to Solve:

1. Scan from Left to Right.
2. If you see a **Number** $\to$ Push to Stack.
3. If you see an **Operator** (+, *, -) $\to$ Pop the top 2 numbers, calculate, and Push result back.

Example: `5 3 + 2 *`

1. Push **5**, Push **3**. (Stack: 5, 3)
2. See `+`. Pop 3 and 5. Calculate $5+3 = 8$. Push **8**. (Stack: 8)
3. Push **2**. (Stack: 8, 2)
4. See `*`. Pop 2 and 8. Calculate $8 \times 2 = 16$. Push **16**.
5. **Answer:** 16.

Topic 4: Trees (The Big Scorer - 4 Marks)

This is the most important part. Master this logic.

Concept: A hierarchy. **Root** is the top. **Leaf** is a node with no children.

Binary Search Tree (BST):

- **Rule:** Smaller value goes **Left**. Larger value goes **Right**.
- **Trick:** If you do an **Inorder Traversal** of a BST, you get the numbers in **Sorted Order** (Ascending).

The 3 Traversals (MEMORIZE THIS)

You will be given a tree diagram and asked to write the sequence.

1. **Preorder: Root** $\to$ Left $\to$ Right. (Root is First).
2. **Inorder:** Left $\to$ **Root** $\to$ Right. (Root is Middle).
3. **Postorder:** Left $\to$ Right $\to$ **Root**. (Root is Last).

The "Flag Method" (Cheat Code to solve in 10 seconds):

Draw the tree on paper. Trace the outline of the tree starting from the left of the root.

- For **Preorder**: Mark a dot on the **Left** of every node. Read dots as you pass them.
- For **Inorder**: Mark a dot on the **Bottom** of every node.
- For **Postorder**: Mark a dot on the **Right** of every node.

Topic 5: Sorting & Searching

A. Binary Search

- **Concept:** Finding a page in a book. You open the middle. If the page is smaller, you ignore the right half. Repeat.
- **Condition:** The list **MUST BE SORTED** first.
- **Complexity:** $O(\log n)$.

B. Sorting Algorithms (One-Liners)

- **Bubble Sort:** Swaps adjacent elements. Slow ($O(n^2)$).
- **Quick Sort:** Fastest in practice. "Divide and Conquer". Worst case $O(n^2)$ but usually $O(n \log n)$.
- **Merge Sort:** Always $O(n \log n)$. Uses extra memory. Stable.

Final Cheat Sheet for Exam Day

If the question asks...	Your Answer is...
Recursion / Undo / Backtracking	Stack
Printer Queue / BFS / CPU Scheduling	Queue
Shortest Path	Dijkstra's Algorithm
Detect Loop in List	Floyd's Cycle Finding (Slow/Fast Pointers)
Sorted Output from Tree	Inorder Traversal
Binary Search Requirement	Sorted Array
Worst Sorting Algorithm	Bubble Sort ($O(n^2)$)
Fastest Searching	Hashing ($O(1)$)