Algorithms Foundations

From Stacks to Binary Heaps

Structure Determines Performance

1. Data Structures vs Algorithms

A data structure is a container with guarantees (array, list, stack, queue, BST, heap).

An algorithm is a rule set that assumes those guarantees.

Performance emerges from structure, not magic.

> **Concept Anchor:** Algorithms assume. Containers guarantee.

Review Questions:

- Why does binary search fail on linked lists?
- What structural guarantee does a heap provide?

---

2. Stacks and Queues

Stack: LIFO structure with O(1) push and pop.

Queue: FIFO structure with O(1) enqueue and dequeue.

Stacks drive DFS and recursion; queues drive BFS.

> **Concept Anchor:** Stacks restrict access; queues enforce fairness.

Review Questions:

- Why must BFS use a queue?
- What happens if DFS uses a queue instead of a stack?

---

3. Trees and Traversals

A tree is a connected, acyclic graph with n-1 edges.

Tree height determines performance.

Preorder, Inorder, Postorder traversals each serve a purpose.

> **Concept Anchor:** Height determines complexity.

Review Questions:

- Why does inorder traversal sort a BST?
- Why is postorder safe for deletion?

---

4. Binary Search Trees

BST property: left < root < right.

Balanced BST: O(log n) operations.

Skewed BST: O(n) operations.

> **Concept Anchor:** A skewed BST is a linked list in disguise.

Review Questions:

- Why does insertion order matter in BSTs?
- What is an inorder successor?

---

5. Graphs and Traversal

Graph defined as G = (V, E).

Adjacency list: O(V + E).

Adjacency matrix: O(V^2).

BFS and DFS both run in O(V + E).

> **Concept Anchor:** Sparse vs dense is about growth rate, not percentage.

Review Questions:

- Why is adjacency matrix inefficient for sparse graphs?
- When is BFS preferred over DFS?

---

6. Binary Heaps

Binary heap is a complete binary tree with heap-order property.

Insert and remove: O(log n).

Peek: O(1).

Heapify: O(n).

> **Concept Anchor:** Heaps guarantee urgency, not total order.

Review Questions:

- Why must heaps be complete?
- Why is heapify O(n)?
