📘 Study Guide: Data Structures

1. What Are Data Structures?

A data structure is a way of organizing, storing, and accessing data to enable efficient operations. Choosing the right structure affects performance, memory use, and algorithm complexity.

---

2. Core Data Structures

Arrays

- **Definition:** Contiguous block of memory; index-based.
- **Strengths:** Fast access O(1).
- **Weaknesses:** Fixed size, expensive inserts/deletes O(n).
- **Use cases:** Static data, lookups, matrices.

Linked Lists (Singly, Doubly, Circular)

- **Definition:** Nodes connected via pointers.
- **Strengths:** Easy inserts/deletes O(1) when node known.
- **Weaknesses:** Slow access O(n).
- **Use cases:** Queues, music playlists, memory-efficient insert-heavy workloads.

Stacks

- **Definition:** LIFO (Last In First Out).
- **Operations:** push, pop, peek.

- **Use cases:** Function calls, undo operations, DFS.

Queues

- **Definition:** FIFO (First In First Out).
- **Variants:** Circular queue, priority queue.
- **Use cases:** Scheduling, buffering, BFS.

---

3. Trees

Binary Trees

- Nodes have ≤ 2 children.

Binary Search Trees (BSTs)

- Left < root < right.
- Search/insert/delete: O(log n) average, O(n) worst (unbalanced).

Balanced Trees

- AVL tree: strict balance, O(log n).
- Red-Black tree: relaxed balance, also O(log n).

Heaps (Binary Heap)

- Complete tree.
- Min/max heap property.
- Insert/delete root: O(log n).

- Use case: priority queues, Dijkstra's.

Tries

- Prefix tree for strings.
- Fast word lookups: O(k) where k = word length.

---

4. Hash Tables

- **Store key–value pairs.**
- Hash function → bucket index.
- Operations: O(1) average, O(n) worst (bad hashing).
- Collision handling: chaining, open addressing.
- Use cases: dictionaries/maps, caching.

---

5. Graphs

- **Components:** vertices, edges.
- **Types:** directed/undirected, weighted/unweighted.
- **Representations:** adjacency list (space-efficient), adjacency matrix (fast edge check).

Graph Traversal

- BFS (shortest paths in unweighted graphs).
- DFS (cycle detection, topological sort).

---

6. Complexity Analysis

Big-O Basics

- O(1), O(log n), O(n), O(n log n), O(n^2)
- Use for runtime + memory comparisons.