UNIT III: Trees

A Comprehensive Study Guide with Real-Time Examples

Tree Terminology

Root: The topmost node with no parent

Leaf/External Node: Node with no children

Internal Node: Node with at least one child

Parent: Node with outgoing edges to other nodes

Child: Node connected to a parent

Sibling: Nodes with the same parent

Height: Number of edges on longest path from node to leaf

Depth: Number of edges from root to node

Subtree: Tree formed by a node and its descendants

Degree: Number of children a node has

Real-Time Example:

Company Organization Chart: A CEO is the root. Department heads are children of the root (at depth 1). Managers under department heads are at depth 2. Individual contributors are leaves. The height of the tree represents the organizational hierarchy levels.

Binary Trees

A tree where each node has at most two children, referred to as the left child and right child.

Types of Binary Trees:

Full Binary Tree: Every node has either 0 or 2 children

Complete Binary Tree: All levels filled except possibly the last, which is filled left to right

Perfect Binary Tree: All internal nodes have two children; all leaves at same level

Balanced Binary Tree: Height difference between left and right subtrees ≤ 1 for all nodes

Degenerate Tree: Each node has at most one child (essentially a linked list)

Real-Time Example:

Huffman Coding Tree: In data compression (used in ZIP files), a binary tree is built where leaf nodes represent characters. The path from root to a character's leaf determines its binary code. Frequently used characters are closer to the root for shorter encoding.

Binary Search Trees (BST)

A binary tree where for every node, all values in its left subtree are smaller, and all values in its right subtree are larger.

Properties:

Search: O(log n) average, O(n) worst case

Insertion: O(log n) average, O(n) worst case

Deletion: O(log n) average, O(n) worst case

In-order Traversal: Gives sorted sequence of elements

Real-Time Example:

File System: Operating systems use BSTs to organize files and directories. When searching for a file, the system quickly navigates left (smaller names) or right (larger names) to find the target efficiently.

Another Real-Time Example:

Database Indexing: SQL databases use BST-like structures (B-trees) for indexing. When you search for a customer record by ID, the database uses the index structure to quickly navigate to the correct block without scanning the entire dataset.

Tree Traversals

Methods to visit all nodes in a tree systematically.

Depth-First Traversals:

1. In-order (Left-Root-Right):

Visit left subtree → Visit root → Visit right subtree

BSTs: Produces sorted sequence

Real-time use: Processing expressions in mathematical notation

2. Pre-order (Root-Left-Right):

Visit root → Visit left subtree → Visit right subtree

Real-time use: Creating a copy of the tree, expression parsing (prefix notation)

3. Post-order (Left-Right-Root):

Visit left subtree → Visit right subtree → Visit root

Real-time use: Deleting a tree, evaluating postfix expressions

Breadth-First Traversal:

Level-order:

Visit nodes level by level, left to right

Uses a queue data structure

Real-time use: Finding shortest path in unweighted graphs, level-wise processing

Real-Time Example:

Compiler Abstract Syntax Tree: When compiling code, the compiler builds an AST and traverses it. Pre-order traversal generates code, post-order evaluates expressions, in-order linearizes the structure.

Heaps

A complete binary tree satisfying the heap property. Used for priority queues and efficient sorting.

Types:

Max Heap: Parent ≥ both children. Root is maximum

Min Heap: Parent ≤ both children. Root is minimum

Key Operations:

Insert: O(log n)

Extract Min/Max: O(log n)

Heapify: O(n)

Real-Time Examples:

1. Priority Queue in Hospitals: Emergency patients are given higher priority. A min heap stores patients by severity level. The most severe patient (minimum key) is always at the root for immediate treatment.

2. Dijkstra's Algorithm: Finds shortest paths in graphs. A min heap stores distances, and the node with the smallest distance is always extracted first for processing.

3. Heap Sort: Uses heap structure to sort n elements in O(n log n) time. Build a heap in O(n) then extract elements n times.

4. Load Balancing: Servers use a min heap to track current loads. New jobs are assigned to the server with minimum load.