Week 6 Study Summary: Binary Trees, Traversals, and Binary Search Trees

Z5007 Programming and Data Structures · Nov 12–14, 2026 · Dr. Innocent Nyalala

1. Binary Tree Traversals and Structural Laws

A binary tree of n nodes and height h ($h = -1$ empty, $h = 0$ single node) satisfies $h + 1 \leq n \leq 2^{h+1} - 1$, and leaves = two_children + 1 for any non-empty binary tree.

- **Preorder (`N, L, R`):** Root first; used to serialize or copy trees.
- **Inorder (`L, N, R`):** Produces sorted ascending keys on any valid BST.
- **Postorder (`L, R, N`):** Children before parent; used to compute subtree heights or free nodes.
- **Level-order:** Uses a FIFO queue to visit nodes layer by layer.

2. Binary Search Tree (BST) Operations and Bounds Validation

Every key in `node.left` is strictly smaller than `node.key`, and every key in `node.right` is at least `node.key`. Search, insert, min/max, range pruning `in_range(lo, hi)`, and delete (using inorder successor replacement for two-child nodes) all follow one root-to-leaf path in $O(h)$ time. Checking `is_bst` requires propagating `[lo, hi)` bounds downward rather than testing immediate children alone.

Why Height Matters

Inserting $1,000$ sorted keys creates a chain of height 999, turning $O(\log n)$ BST search into $O(n)$ linear scan and motivating self-balancing AVL trees in Weeks 7–8.