Non-Comparison Sorting, Binary Trees and Binary Search Trees

Lecture Script, Worked Traces, Worksheet and Solutions
Z5007 Programming and Data Structures, MTech DSAI | Week 7, Nov 16–20, 2026

1. Breaking $n \log n$: Counting Sort and Radix Sort

> Any comparison sort builds a binary decision tree with $n!$ leaves, forcing worst-case height $\Omega(n \log n)$. Counting sort avoids key comparisons by indexing directly into a frequency array of size k, running in $\Theta(n + k)$ time when keys lie in $0..k$. Placing items from right to left keeps duplicates stable so Radix Sort can sort multi-digit integers digit by digit from least significant to most significant in $\Theta(d(n + k))$.

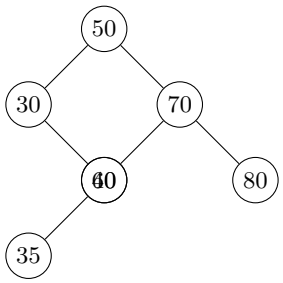
2. Binary Trees and Four Standard Traversals

Every node holds a value plus `left` and `right` references. Depth counts edges from the root down; height counts edges along the longest path to a leaf (empty tree has height -1, leaf has height 0). Traversals visit nodes recursively:

- **Preorder (VLR)**: root first (useful for copying/serializing tree structure).
- **Inorder (LVR)**: left, root, right (yields sorted keys in any valid BST).
- **Postorder (LRV)**: children before parent (bottom-up deletion and subtree heights).
- **Level-order (BFS)**: row by row using a FIFO `deque`.

3. Binary Search Trees (BSTs): Search, Insert and Deletion

The global BST invariant states that for every node x, all keys in x.left are strictly less than x.key and all keys in x.right exceed x.key. Deleting a node falls into three cases: (1) leaf: return `None`; (2) one child: promote that child; (3) two children: replace with the inorder successor (minimum of right subtree) and recursively delete that successor.



Check question. Deleting node 50 above can use either successor 60 or predecessor 40. Which property stays identical under both?

Answer. Both yield valid BSTs with the exact same inorder sequence `30 35 40 60 70 80`, though node balance factors differ.

4. Worksheet (Students Complete Individually)

1. Give preorder, inorder, postorder, and level-order sequences for the tree drawn above (root 50).
2. Draw the BST formed by inserting `25, 15, 40, 10, 20, 35, 50`, then delete `25` using the inorder successor rule.

5. Worksheet Solutions

1. Preorder: `50 30 40 35 70 60 80`; Inorder: `30 35 40 50 60 70 80`; Postorder: `35 40 30 60 80 70 50`; Level-order: `50 30 70 40 60 80 35`.
2. Replacing 25 with successor 35 puts 35 at root with left subtree `15(10, 20)` and right subtree `40(-, 50)`.

References

1. Canning, J., Broder, A. and Lafore, R. (2022). *Data Structures and Algorithms in Python*. Addison-Wesley. Chapters 8–9.
2. Cormen, T. H. et al. (2022). *Introduction to Algorithms*, 4th ed. MIT Press. Chapters 8 and 12.