$\mathrm{DS}_A SSIGNMENT_2$

Syed Shayan Mustufa Rizvi

November 2024

Performance Analysis of Tree Data Structures

Table 1: Performance of Tree Data Structures (in seconds)

Number of Searches	BST (seconds)	AVL (seconds)	B-Trees (seconds)
5,000	24.10	23.40	19.85
10,500	18.25	17.80	17.00
18,500	15.95	15.50	14.02
Average Time	19.43	18.90	16.96

Generalized Asymptotic Analysis (Search, Insertion, Deletion)

- **Binary Search Tree (BST):**
  - **Search:** Best-case $O(\log n)$, worst-case $O(n)$. Observed performance degrades with imbalance.
  - **Insertion:** Similar to search, with rebalancing absent, performance worsens as the tree becomes unbalanced.
  - **Deletion:** Worst-case $O(n)$ if rebalancing is absent. Performance depends on tree height.
- **AVL Tree:**
  - **Search:** Always $O(\log n)$. Empirical results confirm efficiency due to balanced structure.
  - **Insertion:** $O(\log n)$, with slight delays caused by rotations needed to maintain balance.
  - **Deletion:** $O(\log n)$, with rebalancing overhead slightly increasing the runtime compared to B-Trees.

- **B-Trees:**
  - **Search:** Always $O(\log n)$. Observed performance aligns with theoretical predictions, excelling with large datasets.
  - **Insertion:** Efficient $O(\log n)$, optimized for minimizing disk and memory access, leading to superior performance.
  - **Deletion:** Similar to insertion, with $O(\log n)$. Cache optimization ensures deletion is faster than AVL or BST.