2025 AdvDS HW2

B12902036 吳述宇

March 27, 2025

Programming Assignment PA2 – Experiments of BST Upper Bounds and Lower Bounds

Settings

We begin by generating an insertion sequence independent from the access sequences. The insertion sequence is chosen as a random permutation with size $n = 2^{19} = 524288$.

Accessing Sequences

For accessing sequences, my main goal is to select sequences that can highlight performance gaps between various upper bound algorithms. Also, to clearly observe the computational cost, the length of each sequence is $n = 2^{19} = 524288$.

1. Random : This sequence serves as a baseline, providing a neutral, average-case scenario.
2. Monotonic : Designed specifically to expose poor performance of binary search trees lacking self-adjustment mechanisms due to continuous degeneration into a skewed or linear structure.
3. Hotspot : This sequence simulates frequently accessed nodes, reflecting real-world scenarios with temporal locality and skewed distributions.
4. Zigzag : Alternating between large and small keys , this sequence tests how well the upper bounds handle rapid directional shifts in access patterns.
5. Bit-reversal : Known theoretically as a challenging input, thus providing insights into the worst-case adaptiveness of each tree structure.

Upper Bounds

1. Basic : A naive, simple implementation which serves as the worst-case scenario control group, reflecting the fundamental efficiency.
2. Splay : A highly adaptive self-adjusting binary search tree. It is theoretically conjectured to approach dynamic optimality.
3. Tango : A theoretically sophisticated data structure based on the reference-tree model, aiming to approximate the Wilber lower bound. It uses preferred path decompositions to guide tree restructuring.

Lower Bound

1. Wilber–1 : It is used as a fundamental point of reference to estimate the minimum number of accesses (or cost).

Experiment Steps

1. First, generate a random insertion sequence using **generator.py** and save it to **insert.txt**.
2. Then, start the accessing process.
   (a) Generate random / monotonic / hotspot / zigzag / bit-reversal accessing sequence in **generator.py** and save it to **sequence.txt**.
   (b) Run **test.cpp** to evaluate the performance for the sequence of three upper bound trees and compute the Wilber 1 lower bound.
   (c) Use **visual.py** to plot comparisons, rotations, execution time, and show how they compare to Wilber–1.
   (d) Repeat the abvoe steps for all remaining access sequences.

Overview of Results and Evaluation Metrics

The goal of this experiment is to explore how various access patterns affect the relative performance of different upper-bound BST algorithms. The results are primarily presented as sequence-centric bar charts, comparing different tree structures under the same access sequence. In addition, tree-centric views showing how each tree performs across all sequences. Both are provided in the appendix.
On the other hand, the evaluation is based on metrics such as execution time(ms), comparisons / Wilber1, and comparisons / rotations. Among these, the number of comparisons is considered the most reliable indicator of the efficiency, as it is platform-independent and closely tied to algorithmic behavior. In contrast, execution time can vary significantly depending on hardware and compiler optimization.

Explanation and Key Observations

The followings are derived from the sequence-centric bar charts.

1. Across all five access sequences, the Tango Tree consistently achieves the lowest number of comparisons, closely tracking the Wilber I lower bound, which aligns with its theoretical design as a near-optimal BST.
2. The Splay Tree exhibits the highest number of rotations in every case, which reflects its self-adjusting strategy. While this adaptiveness is beneficial in some cases, it may lead to unnecessary restructuring and cost.
3. In the random access scenario, the performance of the Splay Tree is similar to the Basic BST. This indicates that in access patterns lacking strong locality or structure, self-adjustment does not guarantee better performances. In contrast, the Tango Tree outperforms both.

4. For the bit reversal sequence, considered one of the most adversarial cases, the Splay Tree fails to improve performance and behaves similarly to its performance under random access, meaning that the sequence make its adaptiveness not generalize well in the scenario.

5. The monotonic access sequence reveals a significant gap between the upper and lower bounds. The Wilber I bound is extremely low due to the linearity of the access pattern, while all three upper bounds perform significantly worse, and similarly to each other due to tree degeneration.

6. For the hotspot access sequence, the Splay Tree is able to exploit the locality of frequent accesses effectively, resulting in substantially better performance than the Basic BST.

7. In the zigzag sequence, the Splay Tree again outperforms the Basic BST, but lags more significantly behind the Tango Tree compared to the hotspot scenario. This suggests that the Tango Tree is better suited to handling alternations in access patterns.

8. While the Tango Tree achieves the best comparison counts in most cases, it often incurs higher execution time due to its more complex structure maintenance mechanisms. It reveals the difference between theoretic and practical observation.

Conclusion

Despite the implementation complexity of the Tango Tree, it consistently outperforms the other structures in terms of comparison cost, and it is the closest to the Wilber I lower bound. This makes it the most theoretically efficient among the evaluated trees.
On the other hand, the Splay Tree, often cited as a strong candidate for dynamic optimality, exhibits mixed performance in my experiment. While it adapts well to specific patterns such as hotspot sequences, it fails to generalize across more adversarial inputs like bit-reversal.
As for future work, it would be valuable to expand the scope of metrics such as space, incorporate additional BST variants, and explore improved or hardware-friendly implementations.

Tree-Centric Analysis (Additional Content)

This section presents performance observations from each tree structure across all access sequences.

1. Basic BST: The performance remains consistent across all five access sequences. It is insensitive to the access pattern.

2. Splay Tree: Performance degradation is observed under random and bit-reversal sequences. However, for structured sequences such as hotspot or monotonic access, the splay tree is able to leverage locality to improve performance.

3. Tango Tree: The Tango Tree exhibits the most stable performance among all tested structures. Its comparison cost remains consistently close to the Wilber I lower bound across all access patterns.

Appendix

Here are some bar charts about my experiment :

Sequence-centric

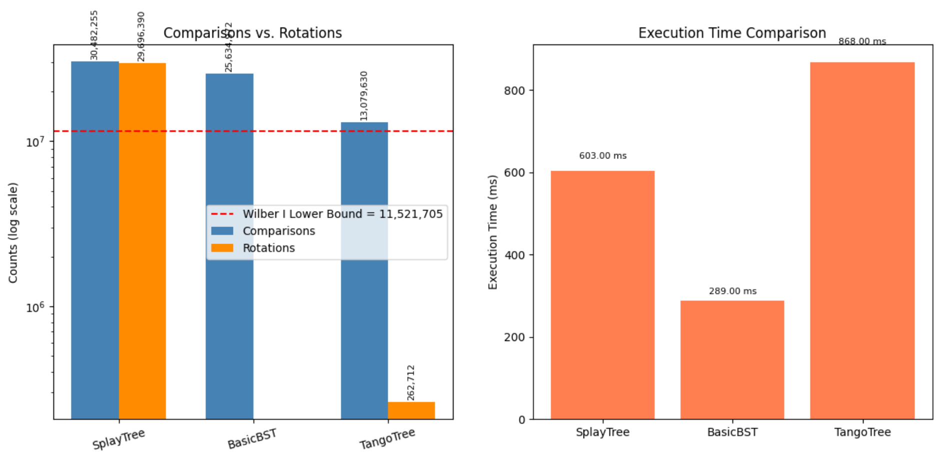

Figure 1: random

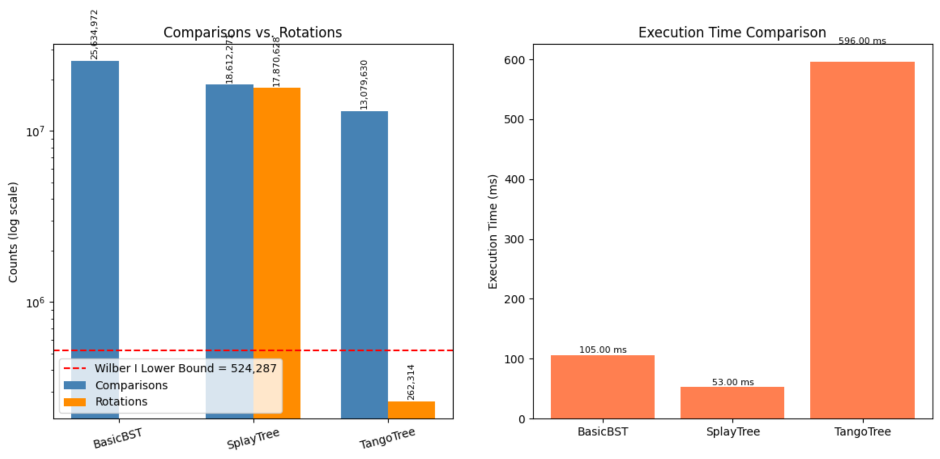

Figure 2: monotonic

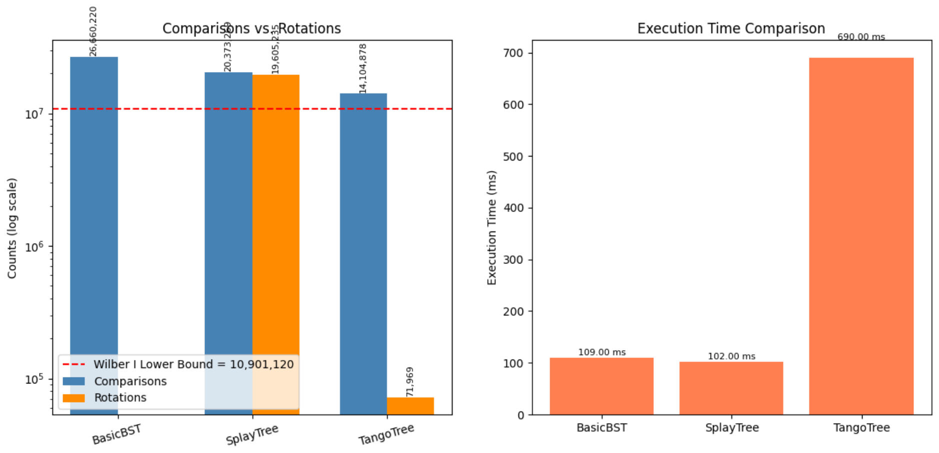

Figure 3: hotspot

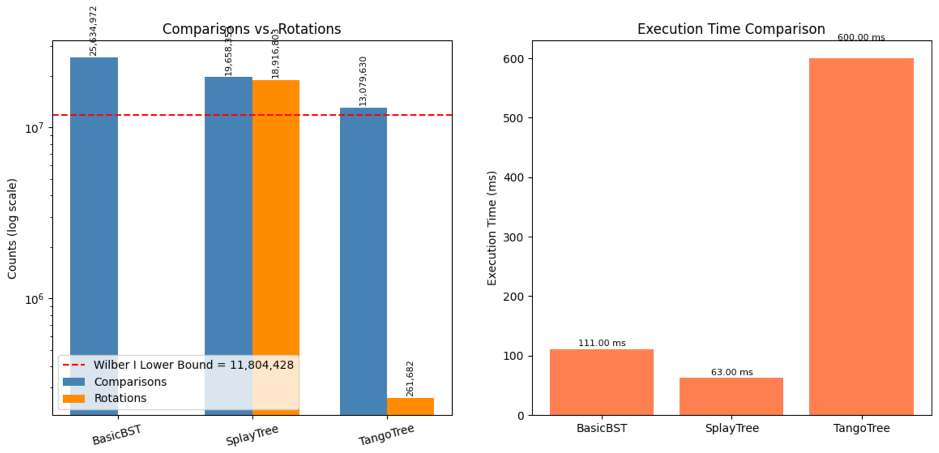

Figure 4: zigzag

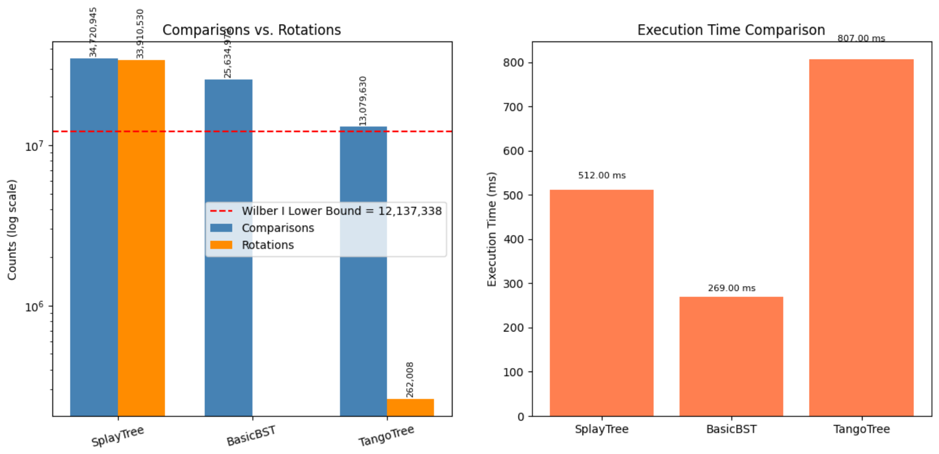

Figure 5: bit reversal

Tree-Centric

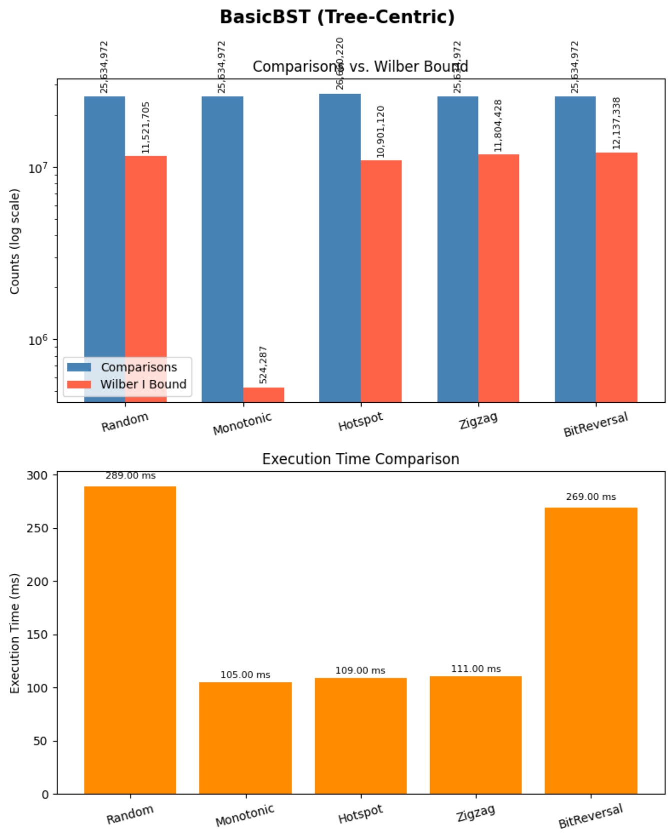

Figure 6: Basic BST

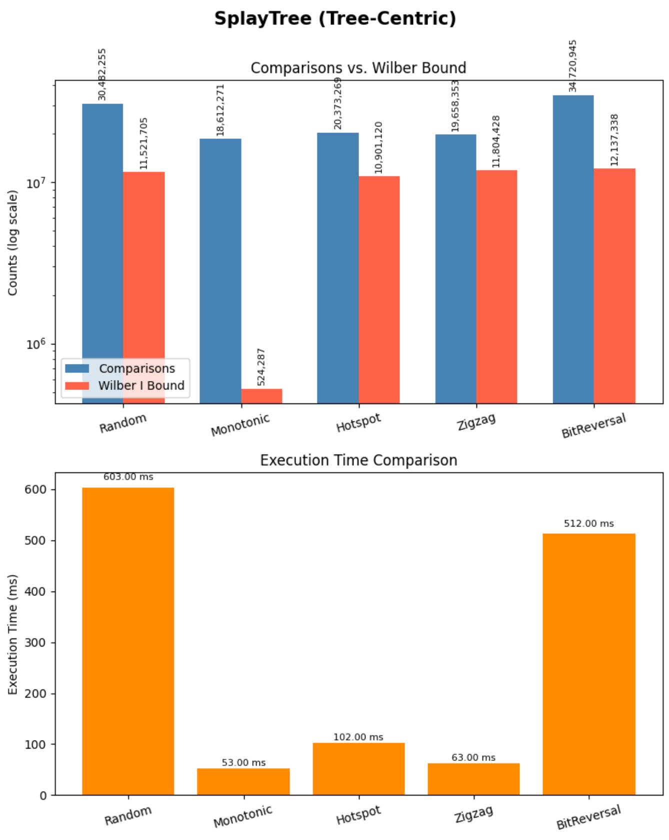

Figure 7: Splay Tree

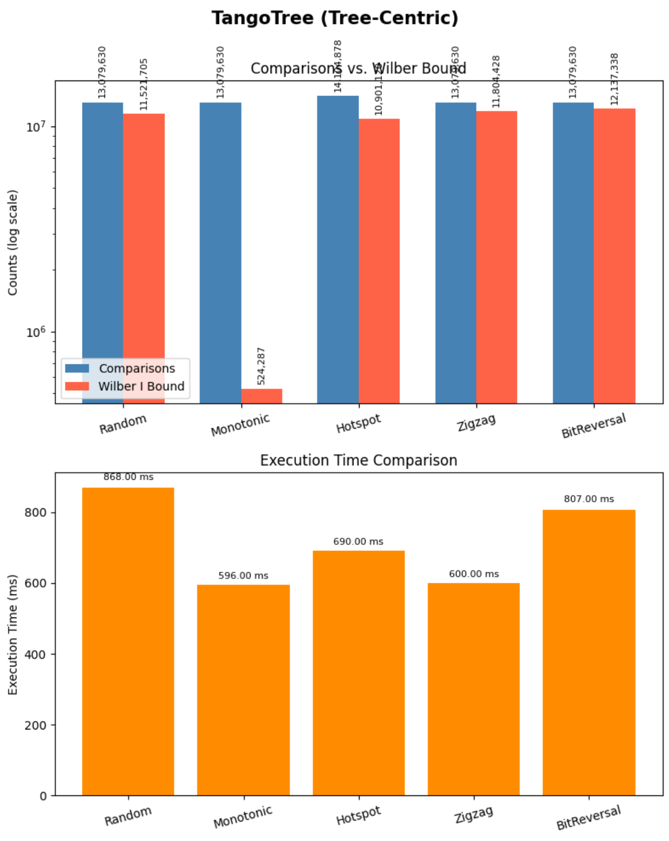

Figure 8: Tango Tree

Thanks **ChatGPT** for helping implementation and visualization.