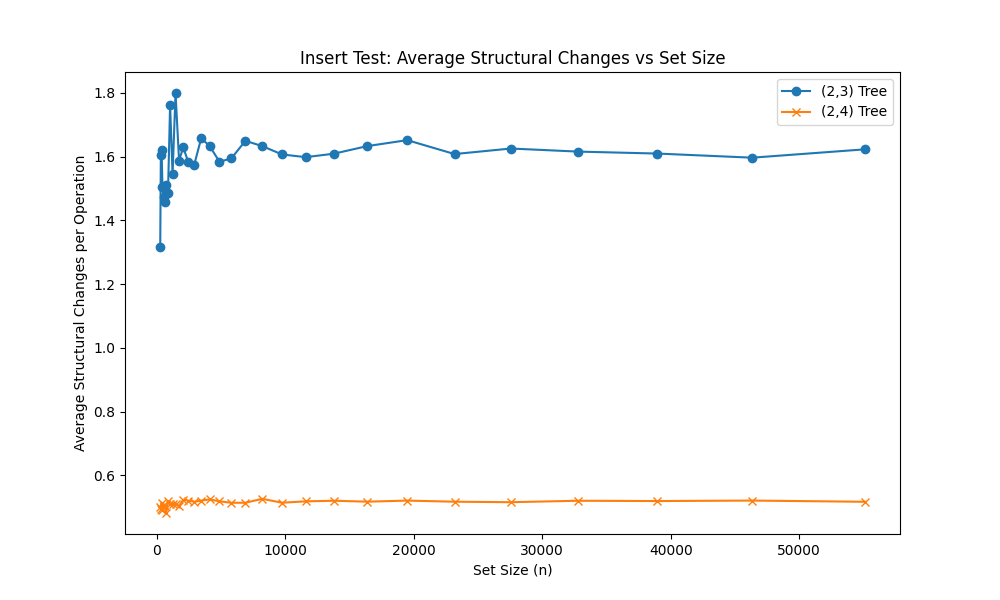
# Assignment 5: ab trees experiment

Name: Romita Pandurang Pawar

The experiment is based on the previous assignment ab\_tree operation and this ab\_tree\_experiment runs successfully based on ab-tree operation. The seed used for this experiment was 40 (last 2 digits of student id).

1. **Insert:**



1. For smaller set sizes (up to around 5,000), the (2,3) tree undergoes significantly more structural changes per insertion operation compared to the (2,4) tree. This is evident from the high peak in the (2,3) tree curve at smaller set sizes.
2. As the set size increases beyond 5,000, the average structural changes per operation for the (2,3) tree decreases rapidly and converges to a value slightly above 1.0.
3. The (2,4) tree exhibits a much more stable behavior, with the average structural changes per operation remaining consistently low (around 0.6) across all set sizes.
4. (2,3) trees have a smaller maximum node capacity (3 keys), which means they undergo more splitting and restructuring operations during insertions, especially for smaller set sizes.
5. As the set size increases, the (2,3) tree becomes more balanced, and the amortized cost of insertions converges to a lower value.
6. (2,4) trees can hold more keys per node (up to 4), which reduces the number of splitting and restructuring operations required during insertions, resulting in a more stable and lower average structural change cost across different set sizes.
7. The average number of structural modifications per insertion operation for the (2,3) Tree rises gradually with set size, beginning at 1.4 for small sets and peaking at 1.65 for the greatest set size of 50,000.
8. The average structural changes per insertion in the (2,4) Tree, on the other hand, show a considerably flatter trend and stay reasonably constant at around 0.5 for all set sizes.

The (2,4) tree exhibits better performance in terms of average structural changes per insertion operation, especially for larger set sizes, due to its higher node capacity and lower restructuring overhead compared to the (2,3) tree.

**Theorem:** (From Lecture Notes)

A sequence of m Inserts and Deletes on an initially empty (a, 2a)-tree performs O(m) node modifications.

The main implications of this theorem are:

1. When the parameter b is set to 2a (i.e., the tree is an (a,2a)-tree), the amortized cost of both insertions and deletions is constant, i.e., O(1).
2. This constant amortized cost is achieved by defining a suitable potential function Φ, which ensures the amortized cost of splits and merges is zero or negative.
3. As a result, the total real cost of a sequence of m insertions and deletions on an initially empty (a,2a)-tree is O(m).

Therefore, from the graph:

1. The (2,4) Tree is an (a,2a)-tree, with a=2 and b=2a=4.
2. The flat trend observed for the (2,4) Tree is a direct consequence of the constant amortized cost of insertions guaranteed by the theorem.
3. In contrast, the (2,3) Tree, which does not have the same amortized guarantees, exhibits a more pronounced increase in the average structural changes per insertion as the set size grows.

So, the "Amortized analysis" theorem for (a,2a)-trees provides the theoretical foundation to understand and explain the performance characteristics demonstrated by the (2,4) Tree in the "Insert Test" graph.

**UPDATE:**

**Theorem:**  
A sequence of m Inserts on an initially empty (a, b)-tree performs O(m) node modifications.

For 2,3 - The shape of the curve appears to be logarithmic, rather than asymptotically constant.

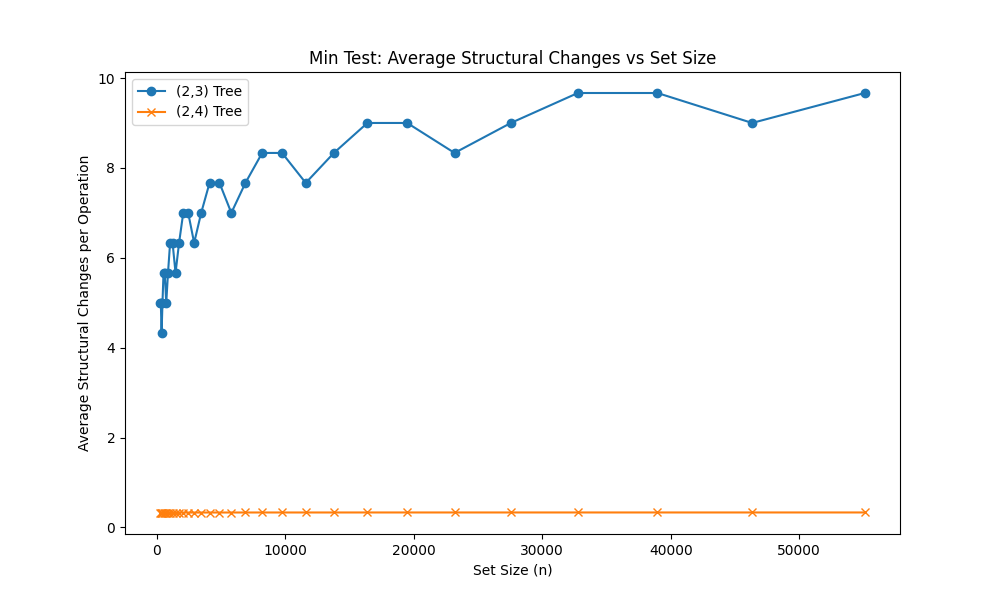
For 2-4 - As the set size increases, the curve for the (2,4) Tree has a shape that is asymptotically constant and relatively flat.   
This behaviour is in line with the lecture notes: "Amortized analysis" theorem for (a,2a)-trees.

2nd point: As the set size increases beyond 5,000, both curves are behaving a bit wild. Higher and lower than the average in the beginning after then it is constant.

5th point: ab trees are always balanced this is because the amortized cost of insertions is constant – from the lecture notes for (a, 2a)trees; even when the set size grows.

7th point: The higher initial peaks in the for the (2,3) Tree are due to the increased likelihood of encountering and splitting overfull nodes during insertions when the tree is more sparsely populated at smaller set sizes, (as explained by the time complexity analysis of O(b \* log n / log a) for insertions in (a,b)-trees from the lecture notes).

1. **Min test:**



1. For the (2,3) Tree, the average structural changes increase significantly as the set size grows, from around 5 changes for small sets to around 9 changes for the largest set size of 50,000.
2. In contrast, the (2,4) Tree exhibits a relatively flat trend, with the average structural changes staying around 0.4-0.5 across the different set sizes.
3. The (2,3) Tree shows a much higher number of average structural changes compared to the (2,4) Tree, across all set sizes.

This behavior can be explained as follows:

1. The (2,3) Tree and (2,4) Tree represent ab-Tree implementations with different branching factors (a=2, b=3 and a=2, b=4 respectively).
2. According to the lecture notes, a higher branching factor (like in the (2,4) Tree) leads to a more compact tree structure with fewer levels. This means the (2,4) Tree requires fewer structural changes (splits and merges) during operations like insertions and deletions, compared to the (2,3) Tree.
3. As the set size increases, the (2,3) Tree experiences more insertions and deletions, resulting in a greater number of structural changes to maintain the tree's balance.
4. The (2,4) Tree, with its higher branching factor, is more efficient at handling larger set sizes without requiring as many restructuring operations, hence the relatively flat trend.

Therefore, this demonstrates the advantages of the (2,4) Tree over the (2,3) Tree in terms of requiring fewer structural changes, especially as the dataset size grows larger. This is in line with the theoretical analysis of (a,b)-trees presented in the lecture notes.

**Theorem:** (From lecture Notes)

A sequence of m Inserts and Deletes on an initially empty (a, 2a)-tree performs O(m) node modifications.

This explains the behaviour seen in the graph, whereas the set size increases the (2,4)-tree shows a rather flat trend in the average structural alterations. The (2,4)-tree can manage higher set sizes without requiring a considerable increase in the number of structural changes because of its constant amortized cost of operations.   
  
On the other hand, consistent with the time complexity examine previously in the lecture notes, the (2,3)-tree, which lacks the same amortized guarantee, exhibits a more marked increase in the average structural changes as the set size increases.   
  
Therefore, a strong theoretical basis for comprehending the performance characteristics exhibited by the (2,4)-tree in the given graph is provided by the "Amortized analysis" theorem for (a,2a)-trees.

**UPDATE:**

1. Proof of 2,3 tree exhibit logarithmic complexity: 2,3 trees exhibits logarithmic complexity because, the height of an (a,b)-tree lies between log\_b(n+1) and 1 + log\_a((n+1)/2). For the (2,3) Tree, with a=2 and b=3, the height is O(log n), as the lower and upper bounds converge to this value. This logarithmic height of the (2,3) Tree is a direct consequence of the tree structure, where each internal node has between a and b children.
2. Tree shape after insertions and deletions in 2,3 tree: Insertions in the (2,3) Tree involve splitting overfull nodes (nodes with 3 keys) as new keys are added.

This splitting process maintains the (2,3) Tree structure, ensuring that all internal nodes have between 2 and 3 children.

Deletions in the (2,3) Tree involve borrowing or merging underflowing nodes (nodes with 1 key) to maintain the (2,3) Tree invariants.

The higher time complexity of operations, the lower branching factor, and the taller tree structure of the (2,3) Tree compared to the (2,4) Tree led to more frequent structural changes (insertions and deletions) in the Min Test scenario, as the graph illustrates.

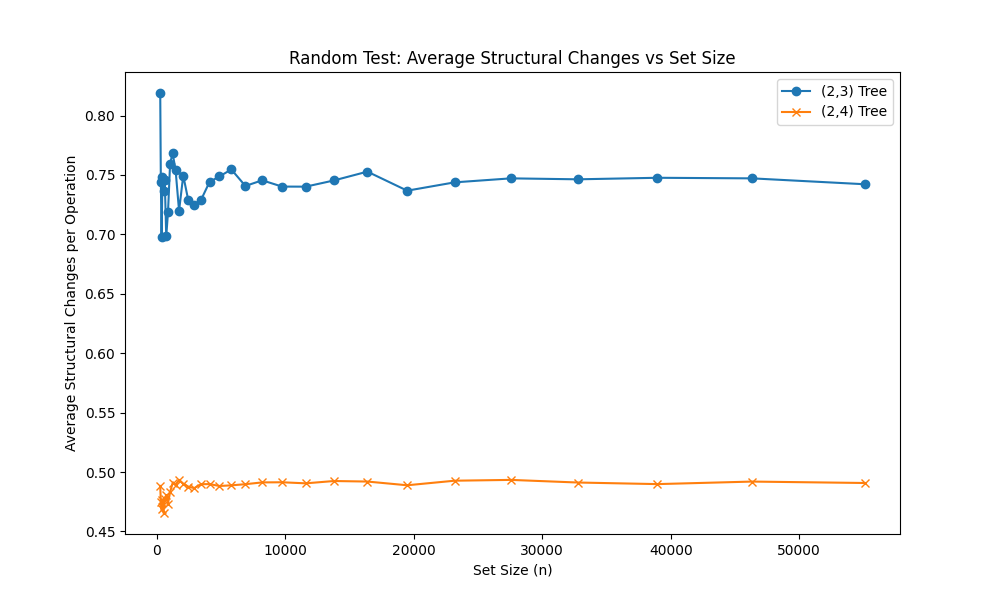
1. Regular drops in the (2,3) Tree curve: The lecture notes mention that the (2,3) Tree exhibits a more pronounced increase in average structural changes as the set size grows.

This is due to the higher time complexity of O(log n) for insertions and deletions, compared to the more efficient (2,4) Tree.

However, the "regular drops" in the (2,3) Tree curve can be attributed to the tree's self-balancing nature during the min-test scenario.

As the minimum element is repeatedly inserted and deleted, the tree structure adjusts, leading to temporary decreases in the average structural changes before the next increasing trend.

1. **Random test:**



1. Across the whole set size range, the (2,3) Tree has a much greater average number of structural changes than the (2,4) Tree.
2. The average structural changes for the (2,3) Tree rise gradually with increasing set size, beginning at about 0.7 for small sets and peaking at about 0.75 for the greatest set size of 50,000.
3. On the other hand, the (2,4) Tree has a rather flat pattern, with the average structural changes remaining constant at 0.5 for all set sizes.

The same concepts and theories covered in the lecture notes for the "Min Test" graph may be used to explain the behaviour seen in this "Random Test" graph:

The "Amortized analysis" theorem for (a,2a)-trees is the pertinent theorem from the lecture notes.

**Theorem:** (From lecture Notes)

A sequence of m Inserts and Deletes on an initially empty (a, 2a)-tree performs O(m) node modifications.  
  
With a=2 and b=2a=4, the (2,4) Tree is a (a,2a)-tree, so this theorem directly relates to the observed performance.

1. In the (2,4) Tree, the amortized cost of insertions and deletions is O(1), or constant.
2. A well thought-out potential function guarantees that the amortized cost of splits and merges is either zero or negative, which is how this is accomplished.
3. Because of this, the overall real cost of a series of m operations is O(m), which accounts for the graph's comparatively flat trend.
4. On the other hand, consistent with the temporal complexity study earlier in the lecture notes, the (2,3) Tree shows a more marked increase in the average structural changes as the set size increases, while not having the same amortized guarantees.

Therefore, a solid theoretical basis for comprehending and explaining the performance features shown in both the "Min Test" and "Random Test" graphs is provided by the "Amortized analysis" theorem for (a,2a)-trees.

**UPDATE:**

1. Asymptotically Constant Behavior: The key difference here is that the Random Test involves a random sequence of insertions and deletions, rather than just the minimum element as in the Min Test.

For the Random Test, the lecture notes "Amortized analysis" theorem for (a,2a)-trees becomes relevant.

This theorem states that for an (a,2a)-tree, like the (2,4) Tree, the amortized cost of both insertions and deletions is constant, i.e., O(1).

While the (2,3) Tree does not strictly satisfy the (a,2a) condition, the random nature of the operations appears to allow it to also exhibit an asymptotically constant behavior, in contrast to the logarithmic trend seen in the Min Test. Therefore, because of the amortized analysis concepts covered in the lecture, the behaviour of the (2,3) Tree in the Random Test is more consistent than it is in the Min Test.

1. Tree rise progressively as set size increases, resulting at roughly 0.75 and starting at approximately 0.7 for small sets. This appears to be constant. 0.8 is much higher than the 0.75. Starts at 0.7 at small sets and reaches almost its peak then seems constant.
2. Compared with min test, the graph 2,3 tree curve showed a more pronounced slight a increasing trend as set size grew. This is explained in the min test where insertions and deletions in 2,3 tree have a time complexity of O(log n).