Diellor Hoxhaj – Ab-Trees Experiment

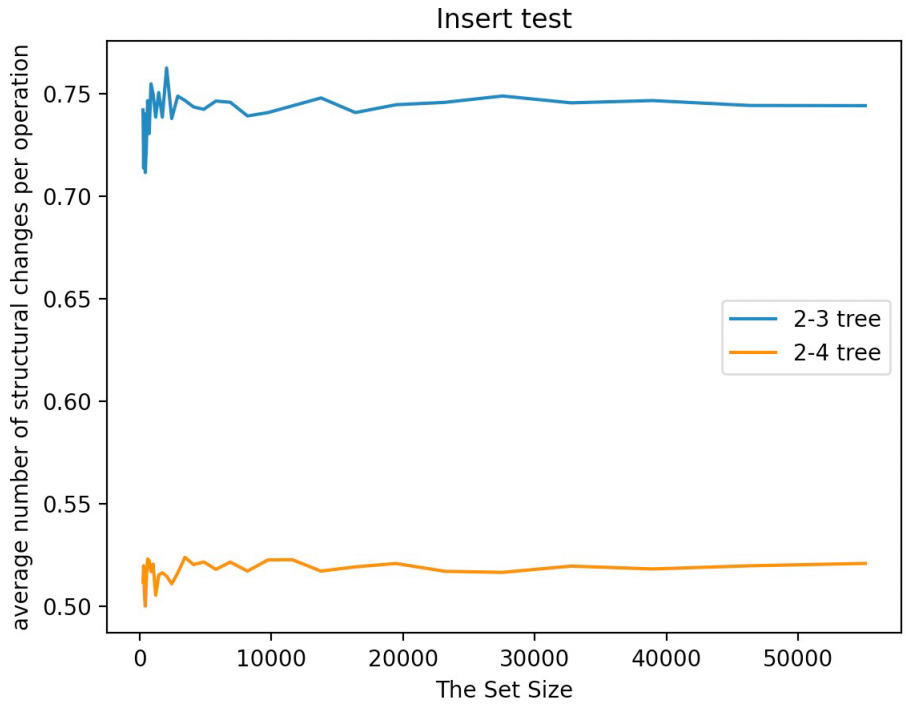

Figure 1 - Insert Test

Insert test: Insert *n* elements in random order.

When we want to insert a key in ab-trees, first we need to find it.
Here we are inserting n elements in random order, in each insert we find the key if exists and if not it ends up in leaf nodes and we perform insertion there (on the leaf nodes).
Finding the right place to insert it compares the given key with all keys of the node, which can be performed in time O(log b) by binary search.

Based on lectures:
In total, we spend time $\Theta(\log n \cdot \log b / \log a)$. If b is polynomial in a the ratio of logarithms is $\Theta(1)$, so the complexity of Find is $\Theta(\log n)$. Insertion visits $\Theta(1)$ nodes on each level and spends $\Theta(b)$ time on each node. This makes $\Theta(b \cdot \log n / \log a)$ time total.

Simply put, insertion cannot perform better than amortized Log N, because it involves a search that can take O(log N) repeatedly.

In an amortized sense, (a, b)-trees cannot perform better than in $\Omega(\log n)$, because all operations involve search, which can take O(log n) repeatedly. We have a theorem from the lecture.
Theorem: A sequence of m Inserts on an initially empty (a, b)-tree performs O(m) node modifications.
Proof: Each Insert performs a (possibly empty) sequence of node splits and then it modifies one node. A split modifies two nodes: one existing and one newly created. Since each split creates a new node and the number of nodes never decreases, the total number of splits is b.

2-4 performs a bit better mostly because while inserting it changes less than 2-3 tree for all insertions.

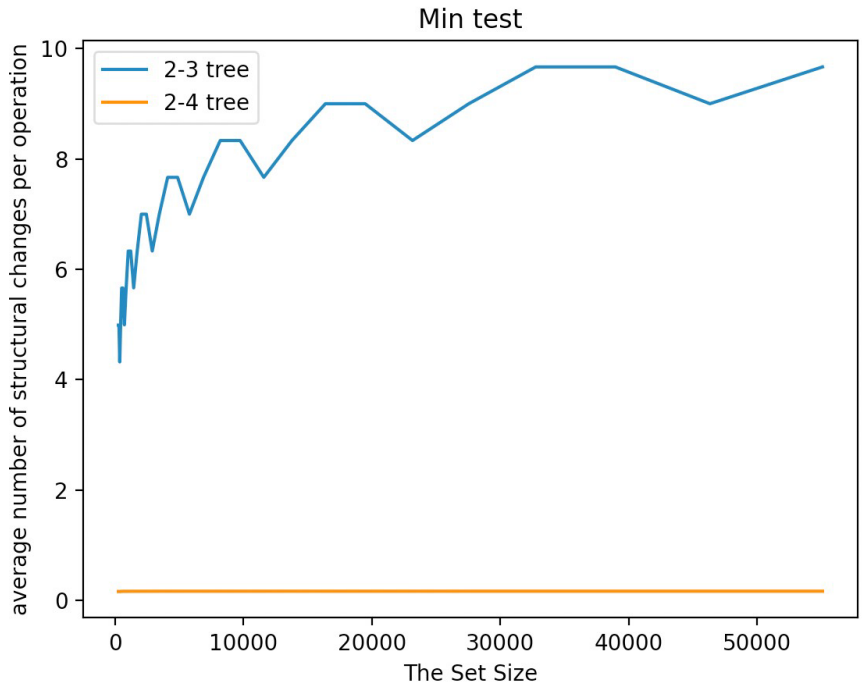

Figure 2 - Min Test

In this test, we can see that 2-3 trees have logarithmic growth and 2-4 tree run constantly.

In the first scenario we can prove this by theorem: "A sequence of m Inserts on an initially empty (a, b)-tree performs O(m) node modifications"

The reason why this is logarithmic is that when we mix it with deletions and insertions again, it might happen that the Insertion forces split up to the root, and in the long sequence of operations we have amortized O(log n) cost each.

With 2-4 trees is not the same case where the amortized number of modified nodes is constant for some choices because the b is 2*a.

We have this in the lecture notes, the theorem states: **Theorem: A sequence of m Inserts and Deletes on an initially empty (a, 2a)-tree performs O(m) node modifications.**

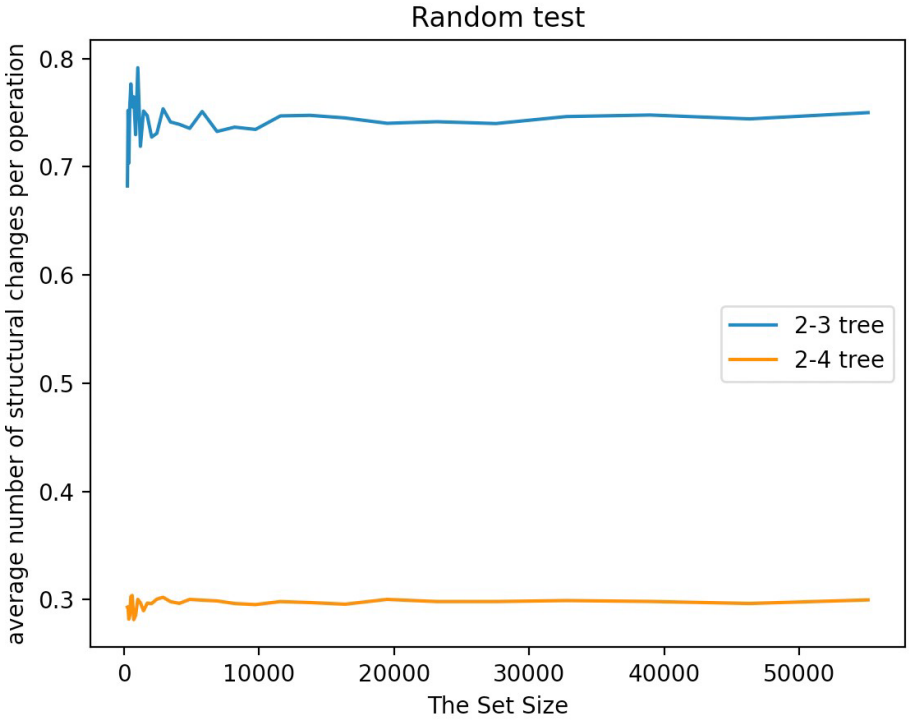

Figure 3 - Random Test

In this test, we see that we have similar results as we had in the first test while inserting n random elements.

Why are they constant?

We have a theory from the lectures:

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

Proof: We define the cost of an operation as the number of nodes it modifies. We will show that there exists a potential Φ such that the amortized cost of splitting and merging with respect to Φ is zero or negative and the amortized cost of the rest of Insert and Delete is constant.