My random seed: 35

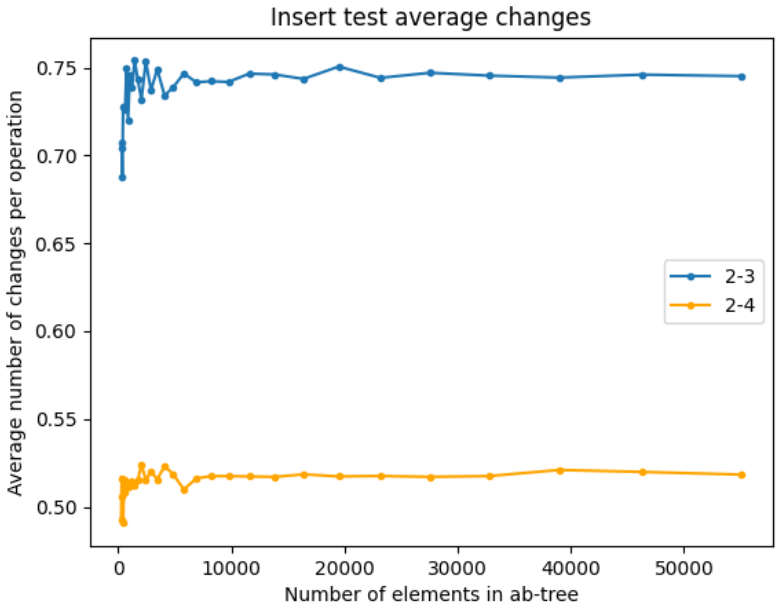

Figure 1: Graph of average number of changes (split, merge) with dependence on number of elements in ab-tree. We insert n elements sequentially.

1 Insert test

As is written in Lecture notes, a sequence of m inserts on an initially empty (a,b)-tree performs $\mathcal{O}(m)$ node modifications. The average number of modifications per operation is therefore constant as can be seen in figure 1.
We can insert 2 values to (2,3)-tree node before we have to split the node to insert the third value to tree. In (2,4)-tree we can insert 3 values before the split of node. The (2,4)-tree therefore has less number of modification per operation than (2,3)-tree.

2 Min test

In figure 2 We can see similar behavior of (2,4)-tree as in the case of insert test. The difference is in the case of (2,3)-tree. It looks like with increasing number of elements the number of modifications per operation grows logarithmically. The explanation is written in the Lecture notes:

"When we start mixing insertions with deletions, the situation becomes more complicated. In a (a,2a-1)-tree, it might happen that Insert(x) forces splitting up to the root and the immediately following Delete(x) undoes all the work and reverts the tree back to the original state. So we can construct an arbitrarily long sequence of operations which have $\Omega(\log n)$ cost each."
We encountered similar problem with flexible arrays: "if the thresholds for stretching and shrinking are too close, the structure tends to oscillate expensively between two states. We cured this by allowing the structure extra “breathing space”.
The same applies to (a,b)-trees. When we increase b to at least 2a, it is sufficient to avoid oscillations and keep the amortized number of changes constant."

This explains the behavior in the figure 2.

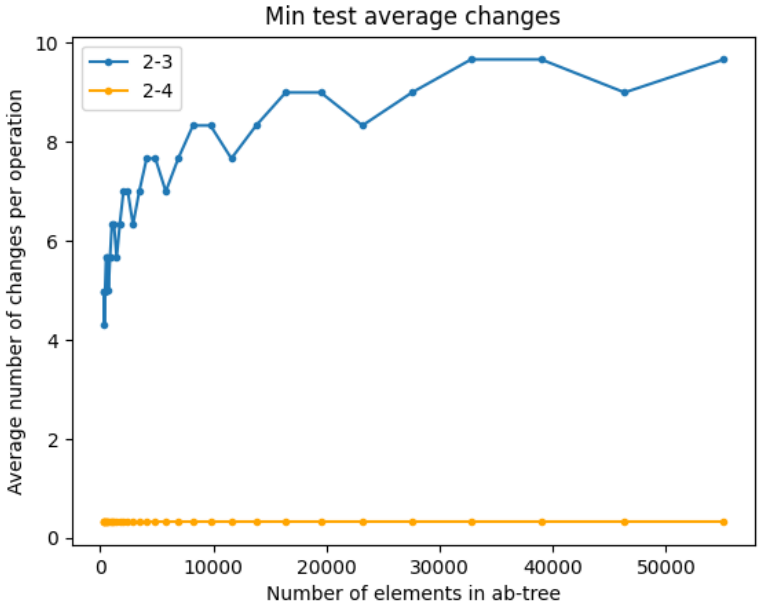

Figure 2: Graph of average number of changes (split, merge) with dependence on number of elements in ab-tree. We first insert n elements sequentially and then n times repeat: remove the minimal element in the tree and then insert it back.

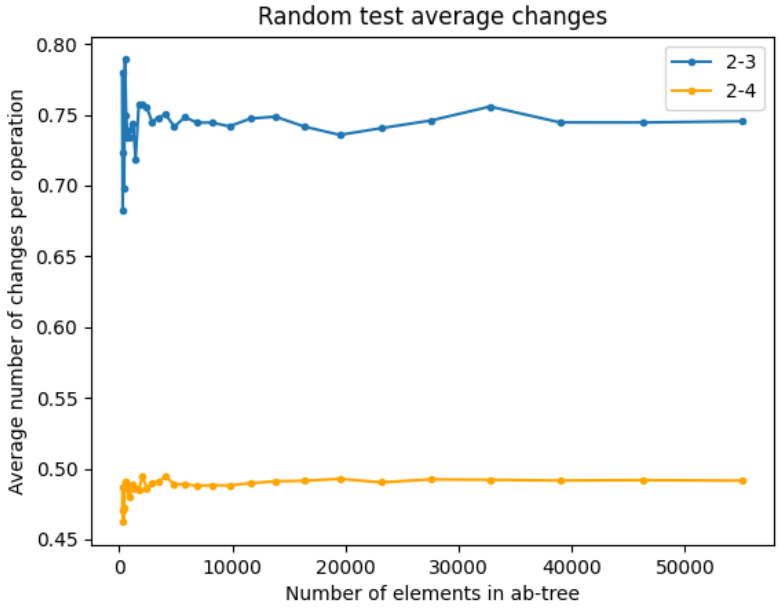

Figure 3: Graph of average number of changes (split, merge) with dependence on number of elements in ab-tree. We first insert n elements sequentially and then n times repeat: remove random element from the tree and then insert random element into the tree. Removed element is always present in the tree and inserted element is always not present in the tree.

3 Random test

We can see in figure 3 that the average number of modifications per operation is in the case of random test similar as in the insert test (fig. 1). Only difference is that the (2,4)-tree in random test has average n. of mod. per op. lower than 0.5 in comparison with insert test where the value was mostly above 0.5. In my opinion, this could be caused by fact that we not only sequentially insert element, but we repeatedly remove and insert elements. I think that in some cases there could be repeated deletions and insertions of values to same node without any merge or split of the given node. In case of insert test, we only insert values so we can perform

only 3 operations on node before it has to be split.