My random seed: 35

1 Sequential test

As we know from lectures, the amortized cost of all operations on splay tree which uses standard rotations (zig-zag, zig-zig, and zig) is $\mathcal{O}(\log n)$. It looks like the average number of rotations per operation grows logarithmicaly with dependece on number of elements in the splay tree in the sequential test (see graph 2) with standard rotations.
However, if we use naive implementation with single rotations (zig), it looks like the average number of rotations grows linearly. I will try to show that the amortized complexity of naive implementation rotation is $\Omega(n)$.

1.1 Complexity of naive implementation

1.1.1 Insert

Each insert of element in sequential order takes a maximum of 1 rotation as it is inserted as right child of root (if it is not a first insert) and then splayed to root. The insert of sequention is done only at the beginning and on average each insert is done in constant number of rotations.

1.1.2 Find

The elements after the insertion form a sequential chain where each element which is not leaf has only left child. The depth of such a tree is therefore $n-1$ (if root is in depth 0). The elements are ordered from the one with smallest value in leaf to the one with largest value in root. The "find phase" then begins with splaying the leaf element to root. This takes $n-1$ rotations but the depth stays $n-1$ and the root element will now have only right child. The next splay therefore takes also $n-1$ rotations but the last rotation is left rotation which leaves root with both left and right children and therefore the new depth of the right subtree of root which contains the elements which have not been splayed yet is $n-2$. We continue like this until we splay all the elements in the right subtree. This leaves us with the same tree forming sequential chain as at the start of the "find phase". In total this takes

$$n-1+\sum_{i=1}^{n-1} i = n-1+\frac{(n-1)*(1+n+1)}{2} = \frac{n^2+3n-4}{2}$$

rotations. Each of n finds on average therefore takes

$$\frac{n^2+3n-4}{2n} = \frac{n}{2}+\frac{3}{2}-\frac{2}{n}$$

rotations. On average each operation in this naive implementation has therefore complexity $\Omega(n)$.

2 Random test

The amortized cost of all operations as in case of sequential test is again $\mathcal{O}(\log n)$. The amortized complexity of naive implementation of random test can in the worst case be same as in the sequential test with naive implementation.
The reason is that in worst case the random generator generate increasing sequence of numbers and we insert the elements in the sequential order. After that in the worst case we again find the elements in the sequential order in each "find phase". But as can be seen in the figure 3, this rare combination did not happen with our random seed. It looks like we got not so badly balanced tree in which the find operation on average does not take much more rotations than the find in standard implementation.

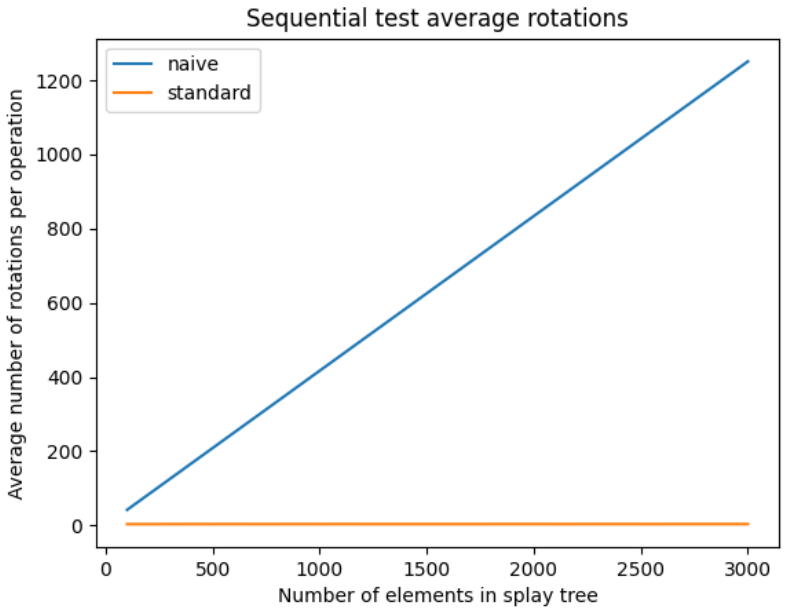

Figure 1: Graph of average number of operations with dependence on number of elements in splay tree. We first insert n elements sequentially and then find them all five times in sequential order.

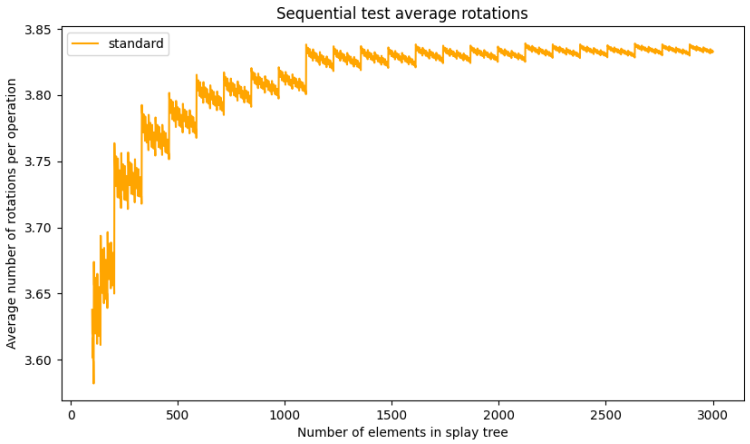

Figure 2: Graph of average number of operations with dependence on number of elements in splay tree. We first insert n elements sequentially and then find them all five times in sequential order.

3 Subset test

First interesting thing that we spot in figure 4 is that naive implementation with same size of subset has lower average number of rotations per operation than standard implementation even when we showed that the amortized complexity of operation with naive implementation is $\Omega(n)$.
So why is the naive implementation as good and even better than standard implementation? My first idea is that naive implementation does not use the "zig-zig" rotation, so it tend to keep the subset closer to root as it does not push nodes above the one being splayed two depths down in "same direction" but first it rotates the parent to be right (left) child of the node being splayed and then it rotates the grandparent to be right (left) child of the node being splayed. In my opinion, the tree will not be as balanced as in the case of standard implementation but the subset will be packed closer to the root.
We can see in figure 5 that the difference between average number of rotations in standard and naive implementation grows with number of lookups we do in both cases on the tree with 1024 nodes and subset of size 100. This growing difference in my opinion means that the double rotation "zig-zig" which is only true difference between naive and standard implementation is the main reason of the standard implementation doing on average more rotations than naive one.

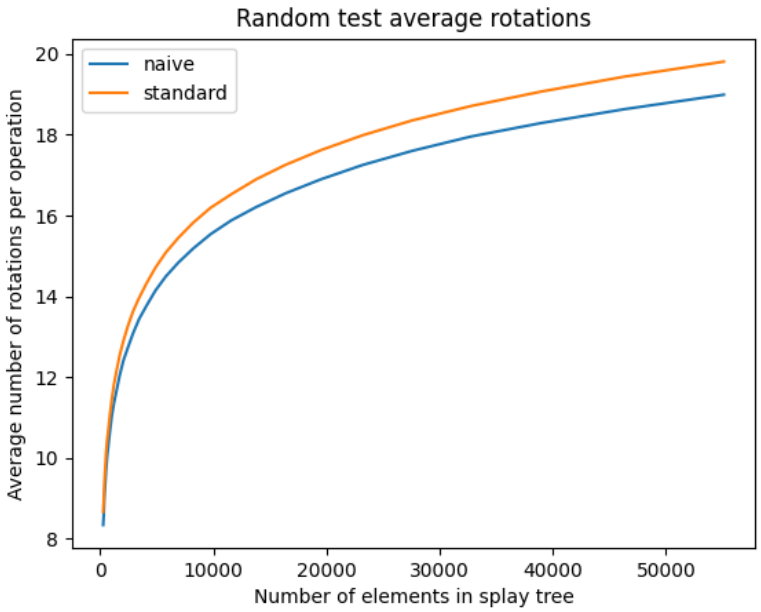

Figure 3: Graph of average number of operations with dependence on number of elements in splay tree. We first insert n elements in random order and then find $5n$ random elements.

Figure 4: Graph of average number of operations with dependence on number of elements in splay tree. We first

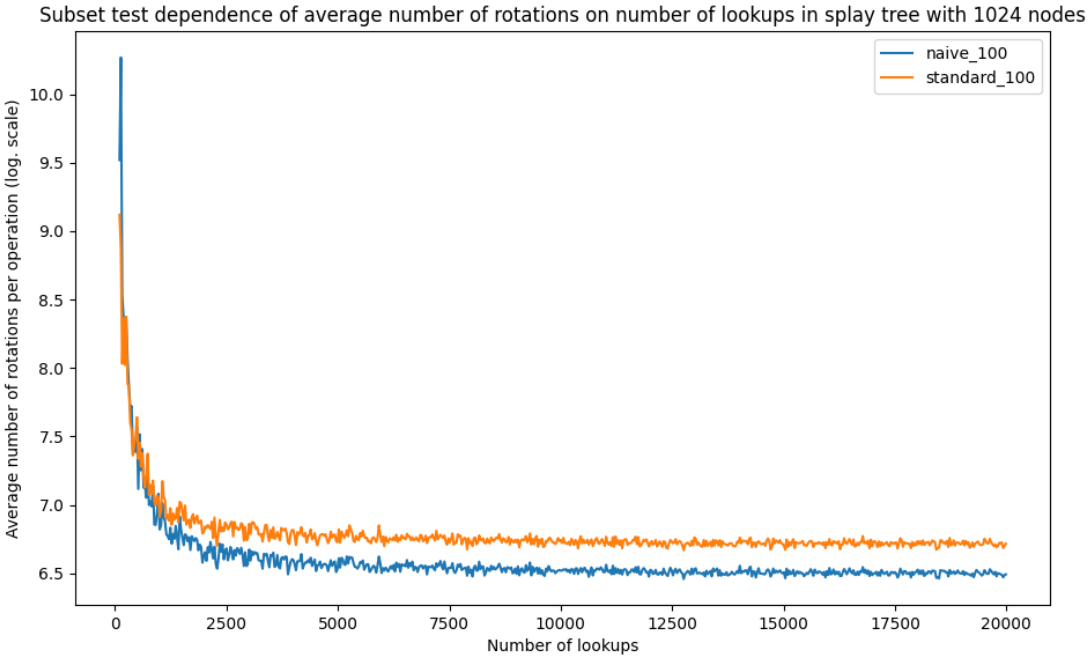

Figure 5: Graph of average number of operations with dependence on number of lookups