Tutorial 4

Data Structures 1, 14. 3. 2025 https://iuuk.mff.cuni.cz/~chmel/2425/ds1en/

Exercise 1 (Splay trees, potentially)
Ensure that you understand the definition of the potential in splay trees and that you understand the main ideas of the amortized analysis of the splay operation.

a) What is the definition of the potential of a splay tree?

b) What is the amortized cost of a rotation (a single and a double)?

c) What is the amortized cost of the full splay operation and how does it follow from the amortized costs of rotations?

d) What is the *real* cost of the splay operation (and what units do we measure it in)?

e) What is the potential of a perfectly balanced BST? (A reasonable upper and lower bound are sufficient, assume the tree has $n = 2^k - 1$ nodes.)

f) What is the potential of a path? (Again, reasonable bounds are sufficient.)

Solution a) Let $T(v)$ be the subtree with root v, its size is $s(v) = |T(v)|$, its rank is $r(v) = \log_2(s(v))$, and the potential is $\Phi = \sum_{v \in V} r(v)$, and we (traditionally) use n to use the number of nodes in the tree.

b) Double $3r'(x) - 3r(x)$, where r' is rank after the operation, r is rank before, and a simple costs $3r'(x) - 3r(x) + 1$.

c) At most $3r'(x) - 3r(x) + 1 \in \mathcal{O}(\log n)$ - this works out, because the sums of rotation costs telescope..

d) The cost is the number of performed rotations, and we measure it in rotations.

e) $\sum_{i=0}^{k-1} 2^i \cdot (\log_2(2^{k-i} - 1)) \leq \sum_{i=0}^{k-1} 2^i \cdot (\log_2(2^{k-i})) = \sum_{i=0}^{k-1} 2^i \cdot (k-i) = \sum_{\ell=1}^{k} \sum_{i=0}^{\ell-1} 2^i = \sum_{\ell=1}^{k} 2^\ell - 1 = 2^{k+1} - 1 - (k+1) = 2^{k+1} - k - 2 = 2(2^k - 1) - k = 2n - \log n$. We also have a trivial lower bound $n/4$, as a complete binary tree has $2^{k-2} \approx n/4$ nodes on the layer just above the leaves.

f) $\sum_{i=1}^{n} \log(i) = \log(n!) \geq \frac{n}{2} \log\left(\frac{n}{2}\right)$, and we can also bound $\log(n!) \leq n \log n$, protože $n! \leq n^n$.

Exercise 2 (Subset theorem)
Recall that the working set theorem of Sleator and Tarjan states the following: Let $x_1, \ldots, x_m$ be a sequence of elements in a BST on n nodes. We set z_i to denote the number of different elements that were accessed after the last access to x_i. Then, the total cost of all the Find(x_i) operations in this order is $\mathcal{O}(n \log n + m + \sum_{i=1}^{m} \log(1+z_i))$.
Using this theorem, prove the "subset theorem": Let T be a tree on the elements $[n]$ and consider a subset $S \subseteq [n]$, where we denote $s := |S|$. Then the queries for elements $x_1, \ldots, x_m \in S$ have total cost $\mathcal{O}(n \log n + m \cdot \log s)$.

Solution
As we only ever query elements in S, it is clear that $z_i \leq s - 1$. Then, we plug in the upper bound on z_i and we get $\mathcal{O}(n \log n + m + \sum_{i=1}^{m} \log(s))$, and this is clearly $\mathcal{O}(n \log n + m \log s)$.

Exercise 3 (Semisplay)
In the zig-zig steps, we could be lazier and instead of performing a double rotation, we could just rotate a single rotation of the top edge. To be precise, in the case LL, let us have three nodes x, y, z such that x is a left child of y and y is a left child of z. Then, when splaying x, we perform a single rotation of the edge zy and we continue by splaying the node y (instead of x). In particular, the originally splayed element might not become the root in the end. The zig-zag step and the final single rotation work in the same way. We call this operation semisplay. Analyze the amortized cost of semisplay (try using a similar analysis as for splay).

Solution
We perform an analysis that is very similar to the standard splay, and we will note that for zig-zag, we actually only need to bound $2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z) \leq 2 + r'(y) + r'(z) - 2r(x) \leq 2(r'(x) - r(x))$, where

the first inequality follows from $r'(x) = r(z), r(x) \leq r(y)$, and the second inequality follows from concavity of the logarithm.
In zig-zig, the potential of the node x does not change, so the cost is $1+r'(y)+r'(z)-r(y)-r(z) \leq 2(r'(y)-r(y)) \leq 2(r'(y) - r(x))$. This telescopes to $2(r'(\text{new root}) - r(x)) + 1$ and the remainder of the analysis works as if we analyzed splay.

Exercise 4 (Extremely lazy splay trees)
Motivated by our laziness that led to the successful operation semisplay, we would like to perform even fewer rotations. What would happen if, when splaying an element x, we would rotate only every second edge on the path from x to the root and gradually change the splayed element as we move up the tree?

Solution
This will not work, as we can build a tree (or, more precisely, a path) of a "zig-zag"shape, with the nodes being $1, n, 2, n-2, 3, n-3, \ldots$ from top to bottom, and then, if n has the right parity, splaying the bottom-most element just changes the shape to $n, 1, n-1, 2, n-2, 3, \ldots$, so the tree does not really change

Exercise 5 (Split and join)
For a splay tree T and a value k, design the operation Split that splits T into two trees T', T'', such that T' contains all nodes with values less than or equal to k, and the tree T'' contains all nodes with values higher than k. If possible, try to make sure that the amortized complexity remains logarithmic.
Next, for splay trees T', T'' (where all values in T' are smaller than all values in T''), design the operation Join(T', T'') that joins the two trees into one (again, while preserving logarithmic amortized complexity).

Solution a) Splay(k) ot Lookup(k), then T' is the root and the root's left subtree, T'' is the right subtree

b) We splay the maximum in T', and then we add T'' as the root's right child.

Bonus exercises

Exercise 6 (Analysing the successor using a potential)
During the first exercise, we showed that using $n-1$ successor operations on any BST, when starting in the node with minimum key, has total cost $\mathcal{O}(n)$. Prove the same result using a potential.

Solution
We define the potential $\Phi(x) := d + c_L(x) - c_R(x)$ where x is the current node, d is the maximum depth of the tree, $c_L(x)$ is the number of left children on the path from the root to x, and $c_R(x)$ is the number of right children on the path from the root to x.
In particular, the cost of finding the leftmost node costs its depth even amortized, but that is not an issue.
Let us consider some cases of applying the successor operation.

1. If x has a right child and y is the successor of x, then $c_R(x)-c_R(y) = -1, c_L(y)-c_L(x) =$ the depth of y under the right c and thus $\Phi(y) - \Phi(x)$ is on the order of the real cost of the operation.

2. If x does not have a right child, we go upwards until we first move into an ancestor from its left child. In that case, for y that is a successor of x, we have $c_R(x) - c_R(y) =$ the number of edges we have climbed up $- 1$ and $c_L(y) - c_L(x) = -1$, which again yields that $\Phi(y) - \Phi(x)$ is on the order of the real cost of operation.

We can also immediately note that the cost of potential is at least zero and at most $2h$.
Thus, except for the first operation running in time h we have all other operations amortized in $\mathcal{O}(1)$, and thus the total complexity is $\mathcal{O}(n + h) = \mathcal{O}(n)$ as $h \leq n$.