Exercise 6.1  Number of different Search Trees.

Let $K_n = \{1, 2, \ldots, n\}$ be a set of keys. Derive a recursive formula for the number of different binary search trees that contain exactly the keys in $K_n$. You do not need to eliminate the recursion.

Exercise 6.2  Dictionary Operations on AVL Trees.

a) Draw the resulting tree if you insert the keys 14, 17, 9, 4, 2, 1, 23, 24, 22 in this order into an initially empty AVL tree.

b) Draw the resulting tree if you remove the key 1 from a).

Exercise 6.3  Number of Rotations when deleting from an AVL Tree.

Consider a binary search tree. A node without successor is called leaf, and every node that is not a leaf is called inner node. Furthermore we define the height of the root as 1, and the height of every other node as the height of its predecessor plus 1. The height of a tree is the maximum occurring height.\(^1\)

a) Describe a recursively defined AVL tree $T_h$ of height $h$, where every inner node has a balance factor of 1. Draw the resulting trees for every $h \in \{3, 4, 5\}$.

b) Let $S(h)$ be the number of nodes of $T_h$. Show by general induction over $h$ that $S(h) \leq 2^h$. Which lower bound follows for the height $h$?

c) Show by general induction over $h$: the minimum height of a leaf in $T_h$ is exactly $\lceil (h+1)/2 \rceil$.

d) Assume that $h$ is odd, and show: deleting the leaf with height $\lceil (h+1)/2 \rceil$ from $T_h$ results in $\Omega(\log n)$ many rotations.

Note: For the proofs b)–c) the principle of general induction must be used. Let $\mathcal{A}(n)$ be a statement for a number $n \in \mathbb{N}$. If, for every $n \in \mathbb{N}$, the validity of all statements $\mathcal{A}(m)$ for $m \in \{1, \ldots, n-1\}$ implies the validity of $\mathcal{A}(n)$, then $\mathcal{A}(n)$ is true for every $n \in \mathbb{N}$.

\[
\forall n \in \mathbb{N} : \left( \forall m \in \{1, \ldots, n-1\} : \mathcal{A}(m) \right) \Rightarrow \mathcal{A}(n) \Rightarrow \forall n \in \mathbb{N} : \mathcal{A}(n). \quad (1)
\]

Thus, general induction allow multiple base cases and inductive hypotheses.

\[^1\]There might exist slightly different definitions in the literature.
Exercise 6.4  Joining AVL Trees.

Let $T_1$ and $T_2$ be two AVL trees that contain the key sets $K_1$ resp. $K_2$. Let the largest key in $K_1$ be smaller than the smallest key in $K_2$, i.e. we have $K_1 \cap K_2 = \emptyset$. Design an algorithm that joins the AVL trees $T_1$ and $T_2$ in time $O(\log n)$, i.e. that computes an AVL tree containing exactly the key set $K_1 \cup K_2$.

Note: The challenge is to ensure that the resulting tree is again an AVL tree (and not an arbitrary unbalanced search tree).

Hand-in: until Wednesday, 2nd April 2014.