8. Problem set on

Algorithms, Data Structures and Data Abstraction WS 18/19

Wolfgang Mulzer

Due on 14. December 2018, 10:00, in the respective TA's mailbox

Problem 1 Binary Search Trees *10 Points*

(a) Insert the keys A, L, P, D, R, E, I, X, Y, Z in this order into a binary search tree that is initially empty. Next, delete the keys Z, A, L. Draw the tree after each insertion and after each deletion.

(b) Let T_1 and T_2 be two binary search trees that both store the same set S of n entries. Show that there is a sequence of at most $2n$ rotations that converts T_1 into T_2.

Problem 2 AVL-Trees *10 Points*

(a) Insert the keys A, L, P, D, R, E, I, F, O, M, N in this order into an AVL-tree that is initially empty. Next, delete A, M, R. Draw the tree after each insertion and after each deletion. Furthermore, show the individual rotations, and annotate the nodes of the trees with their heights.

(b) Show that when inserting an element into an AVL-tree, we perform at most one (single or double) rotation. Does this also hold for a deletion? Justify your answer.

Problem 3 Weight-Balanced Trees *10 Points*

Let T be a binary search tree. We define a *weight function* w on T as follows: we set $w(\perp) = 1$. Let v be a node in T, with left subtree v_ℓ and right subtree v_r. Then, we set $w(v) = 1 + w(v_\ell) + w(v_r)$.

We call T *weight-balanced*, if for every node v in T, the following holds: let v_ℓ be the left and v_r be the right subtree of v. Then, we have $w(v_\ell) \geq w(v)/10$ and $w(v_r) \geq w(v)/10$.

Show that any weight-balanced tree with n nodes has height $O(\log n)$.

Hint: Use induction. Argue first that $w(v_\ell) \leq \frac{9}{10} w(v)$ and $w(v_r) \leq \frac{9}{10} w(v)$.

Problem 4 Treaps *voluntary, 10 additional points*

A treap is a cross between a **tr**ee and a h**eap**. The nodes of a tree store two items (in addition to the values): a *key* and a *priority*. With respect to the keys, the treap T has the binary search tree property. With respect to the priorities, the treap T has the min-heap property. More precisely, for every node v in T, we have that (i)

all nodes in the left subtree of v store smaller keys than v; (ii) all nodes in the right subtree of v store larger keys than v; (iii) the priorities of the children of v are larger than the priority of v.

(a) Draw a treap for the following sequence of key-priority pairs: (E, 23), (K, 65), (H, 5), (B, 7), (I, 73), (A, 10), (G, 4).

(b) Show that for any sequence of key-priority pairs where all keys and all priorities are pairwise distinct, the tree is uniquely determined.

(c) Provide pseudocode for inserting and for deleting a key-priority pair into/from a treap.

Hing: First, proceed as in a binary search tree. Then, show how to use single rotations to restore the min-heap property.

Remark: One can show that the expected height of a treap is $O(\log n)$, provided that the priorities are generated uniformly at random from the interval $(0, 1)$.