Exercise sheet 4.

Data structures SoSe 2020

László Kozma, Katharina Klost

Due 12:00, May 22th, 2020

Exercise 1 Redundant counter *3+3+4 Points*

Recall the redundant counter from the lecture that uses digits $\{0, 1, 2\}$ and that can be incremented in actual (worst-case) time $O(1)$.

(a) Count from 1 to 20 using the increment method discussed in lecture for this counter. (Recall: the last digit cannot be 2, and between any two 2s there must be at least one 0.)

(b) Describe how two numbers can be efficiently added in this representation (so that the rules are enforced).

(c) Modify the counter so that it also supports a *decrement* operation in time $O(1)$. Describe which digits you use, what rules the representation must satisfy and sketch how an increment and decrement operation can be implemented.

Exercise 2 Binary search trees recap *3+3+2 Points*

(a) The *depth* of a node x in a binary search tree is the number of edges on the path from x to the root of the tree. Define the *subtree-size* of a node x to be the number of nodes in the subtree rooted at x, not counting x itself (note that this differs slightly from the definition in the lecture). The subtree-size of a leaf is thus 0 and the subtree-size of the root is one less than the number of nodes in the tree. Show that in any binary search tree, the average node-depth over all nodes equals the average subtree-size over all nodes. (Check it on some small examples first.)

(b) Recall the rotation operation in binary search trees. Show that given two arbitrary binary search trees T_1 and T_2 with nodes $\{1, \ldots, n\}$, there is a sequence of at most $2n$ rotations that transforms T_1 into T_2.
Hint: Rotate to some canonical state.

(c) A *treap* or *Cartesian tree* is a binary tree in which every node stores a *pair* of values. The nodes of the treap satisfy the binary search tree order with respect to the first value of each pair, and the (min)heap-order with respect to the second value of each pair. Construct a treap with the following pairs of values: $(3, 5), (1, 4), (2, 8), (9, 1), (8, 3), (6, 2), (4, 7), (5, 9), (7, 6)$. Is the treap unique?

Exercise 3 Augmented tree 4 *Points*

We would like to store a dynamic set of points in $\mathbb{R}^2$, supporting the operations of adding a point to the set, deleting a point, and *top-query*(a, b), that reports the point in the set with maximal y-coordinate, among those points whose x-coordinate is in the interval $[a, b]$. We assume that no two points have the same x- or y- coordinate.

Design a data structure that implements all three operations in $O(\log n)$ time, where n is the number of points currently stored. It is sufficient to sketch the details. In particular, if you use augmented trees, describe:

(1) what data you need to store,

(2) how it can be maintained during updates, and

(3) how it can be used to serve the given queries.

Total: 22 points. Have fun with the solutions!