AD – Assignment 4

Binary search trees

Søren Dahlgaard

Hand-in: March 14th

Should be done in groups of 2–3 students.

1 Motivation

The goal of this assignment is to familiarize the student with binary search trees. We will see how we can modify the standard binary search trees from CLRS in order to tackle other problems.

2 Problem statement

The CPS Brewing Company would like to know which of their beers are doing well and which are not. To do this, they wish to investigate the revenue generated by each different beer. As an example they might want to know what the median profit is among all the beers or among some group of beers (e.g. all lager beers).

The CPSBC wants you to develop a data structure that supports the following operations:

- Insert a key.
- Delete a key.
- Query the kth smallest key in the data structure.

An external consultant tells you that this can be accomplished with binary search trees, but you might have to modify the ones from CLRS.

3 Assignment

Consider the node types used in CLRS chapters 12 and 13. We will add an extra field $x.size$, which denotes the size of the subtree rooted in x. The size of a subtree is the number of nodes in that subtree. In order to maintain this field a few things need to be changed.

Task 1: Using the size field described above, give pseudocode for a procedure `Get-kth-Key`(x, k), which returns the kth smallest key in the binary search tree (BST) rooted in x.

If k is not positive or bigger than the number of elements in the tree, your function should return **Nil**.

Task 2: Prove the correctness of your algorithm in task 1.

Hint: You may do this using induction over the tree size.

Task 3: Modify the pseudocode of `Left-Rotate` (CLRS, Fig. 13.3) such that it correctly updates the size field. You do not have to include the entire pseudocode – just state the necessary changes in a readable way.

Task 4: Modify the pseudocode of `RB-Insert` and `RB-Insert-Fixup` (CLRS, p. 315-316) such that the size field is updated. You can assume that the rotate functions update the size field correctly (this is addressed in Task 3 for `Left-Rotate`). You do not have to include the entire pseudocode – just state the necessary changes in a readable way.

Task 5: What is the worst-case running time of `RB-Insert` now? What are the worst-case running times of the three operations discussed in Section 2?

Task 6: Consider a sequence of n insertions/deletions and m queries (that is calls to `Get-kth-Key`) intertwined arbitrarily. What is the worst-case running time of handling such a sequence with your algorithm?