TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Spring Semester, 2000/2001

Homework 3, May 16, 2001

Due on May 30.

Biased search trees

1. Consider the locally biased search trees defined in class.
Define precisely a split operation that splits a tree with root x at a leaf y which is in the tree. The result of the split should be a biased search tree containing the items smaller than y, the item y, and a biased search tree containing items greater than y. You may use catenation as defined in class.
Analyze the amortized time of the split operation that you have defined. Doing it right should give you a split operation with amortized time $O(s(x) - s(y)) = O(\log(W/w_i))$ (while preserving the amortized time of $O(|s(v) - s(u)|)$ for catenating two trees with roots v and u). The sizes are as defined in class.

Splay trees

2. We define the following variation on the splay algorithm. This variation looks 3 steps (edges) towards the root from the node x and applies one of the rules in Figure 1 (or their mirror image) if possible. If it is not possible to apply one of the rules in Figure 1 we apply one of the regular zig-zig, zig-zag, or zig rules (Note that zig or zig-zig would apply only if x is at distance 1 or 2 from the root, respectively).

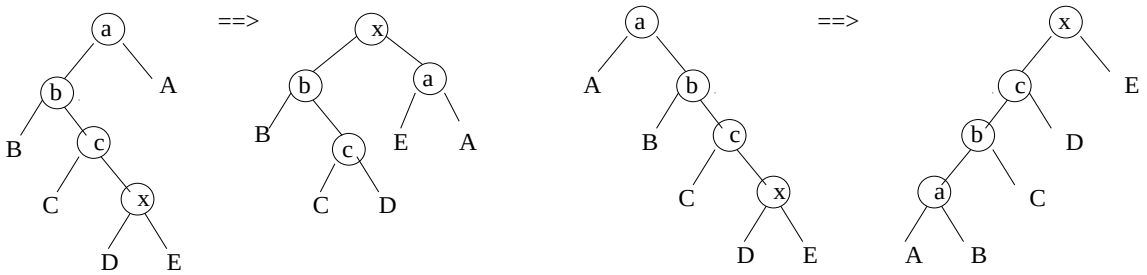

Figure 1: Splay cases in question (3)

Prove that the access lemma holds for this variation as well (with a different constant).

3. Assume that the dynamic optimality is true and prove that to splay the nodes of a binary tree one by one in symmetric order takes $O(n)$ time where n is the number of nodes in the tree.

Dynamic trees

4. Show how to extend dynamic trees to support an operation called $evert(v)$. This operation should change the root of the tree containing v to be v.

5. On a set of n nodes we perform a sequence of operations, each of which is one of the followings.
(a) $insert(u, v)$: Adds an edge between u and v.
(b) $delete - oldest$: Removes the edge that was inserted first among the edges currently in the graph.
(c) $connected(u, v)$: Answers true if there is a path from u to v in the current graph. Otherwise answers false.

Describe a data structure that supports these operations and analyze its performance. (A structure supporting each operation in $O(\log n)$ amortized time would receive maximum score.)