TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Spring Semester, 2003/2004

Homework 3, May 3, 2004

Due on Monday May 17. Don't forget to keep a copy of the homework.

1. Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of distinct items totally ordered such that $a_i < a_j$ iff $i < j$. Assume that over a long sequence of m accesses to items in A, a_i is accessed $q(i) \geq 1$ times. Describe an algorithm to find a binary search tree T where each item a_i resides in a leaf of T such that if we serve the accesses using T, each time going from the root to the corresponding leaf, the total time its takes to serve the whole sequence is minimized. Prove
1) that your algorithm indeed constructs a tree that minimizes the total access time. 2) As tight upper bound as you can, on the total time it takes to serve the sequence using T. 3) An upper bound on the running time of your algorithm for constructing T. (Any polynomial time algorithm would be fine.)

2. Consider the dynamic graph connectivity problem **on a forest** where we allow only two operations:
a) *connected*(v, w) that returns "yes" if and only if v and w are in the same connected component, and
b) *delete*(v, w) that deletes the edge (v, w) from the forest.
Assume we perform m such operations on a forest with n nodes. Describe an efficient (in an amortized sense) algorithm for the problem. (Note that from the material presented in class easily follows a solution that runs in $O(m \log n)$ time, but better solutions exist in particular if $m >> n$.)

3. Consider the move to root heuristic on a tree containing the integers 1 to n in symmetric order (i.e. for every node v all items in the left subtree are smaller than the item at v, and the item at v is smaller than all items at the right subtree). Recall that in the move to root heuristic, after you access an item you perform rotations bottom up on the edge between the accessed item to its parent until the item is at the root. Assume $i < j$.
a) Prove that i is an ancestor of j if the most recent request for i has occured since the most recent request for any of $i + 1, \ldots, j$. Similarly, j is an ancestor of i if the most recent request for j has occured since the most recent one for any of $i, \ldots, j - 1$.
b) Prove that any tree T containing these items in symmetric order can be produced from any other tree containing these items in symmetric order by a sequence of at most n requests under the move to root heuristics.
c) For any m sufficiently large show a sequence of accesses such that serving it with the move to root heuristics takes $\Omega(n)$ time on average per access. (The initial tree is arbitrary.)

4. 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).

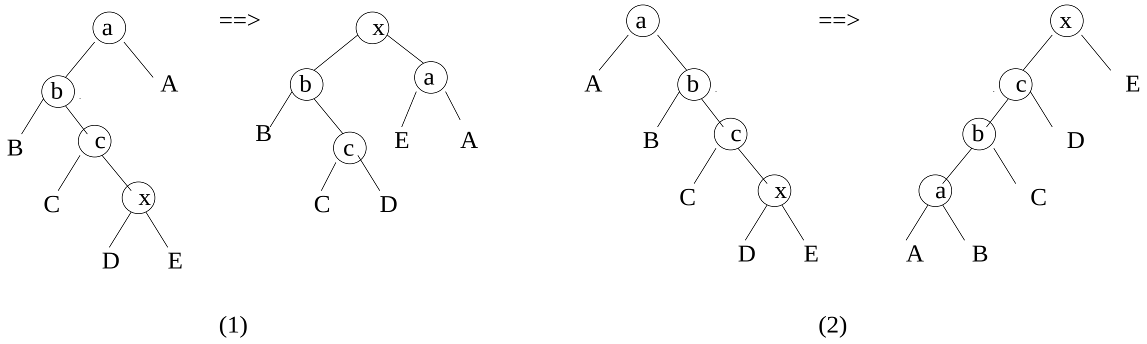

Figure 1: Splay cases in question (3)

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

5. 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.

6. 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.)