TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Fall Semester, 2004/2005

Homework 1, October 25, 2004

Due on Friday November 13. Please keep a copy of your homework ! Hand in your homework via email or to me personally, make sure I mark up that you handed it in. You may be able to find some of the solutions (or ideas leading to them) in the literature, but I think it is more fun to first try by yourself. The questions are not very hard and sometimes there may be more than one interesting solution.

1. A double ended queue (deque) is a data structure which supports insertions and deletions of items from both sides. The operations push and pop, insert and delete an item, respectively, from one side of the deque. The operations inject and eject, insert and delete an item, respectively, from the other side of the deque. Describe an implementation of a deque via stacks (where a stack supports only push and pop). Simulate each operation on the deque by a sequence of operations on the stacks. Your implementation should require $O(1)$ amortized time per operation and $O(n)$ space. Prove that indeed your data structure achieves this amortized time bound.

2. Prove that a sequence of m intermixed insertions and deletions (where the point of insertion or the item to be deleted are given) into an initial 2-4 tree with n nodes takes $O(m+n)$ time.

3. A 2-3 tree is defined similarly to 2-4 trees except that internal nodes are restricted to have only 2 or 3 children. We implement insertion by adding the new leaf as a child of the appropriate node x. If node x becomes a 4-node we split it into two 2-nodes and we continue the splittings until we hit a 2-node that becomes a 3-node. We implement deletion by removing the corresponding leaf. If its parent was a 3-node it becomes a 2-node and the operation is complete. Otherwise the parent, say y, becomes a 1-node. If y has a 3-node as a sibling it borrows a child from its sibling. If y does not have a sibling which is a 3-node we fuse it with its sibling and continue in the same manner at the parent of y if it becomes a 1-node.

Show a sequence of m insertions and deletions to an n node tree, where each insertion or deletion requires $\Omega(\log n)$ time even if the point of insertion or the item to be deleted are known.

4. 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.)

5. 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 $O(r(x) - r(y)) = O(\log(W/w_i))$ (while preserving the amortized time of $O(|r(v) - r(u)|)$ for catenating two trees with roots v and u). The ranks are as defined in class.