TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Spring Semester, 2001/2002

Homework 1, March 20, 2002

Due on April 10. 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.

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 two stacks (where a stack supports only push and pop). Simulate each operation on the deque by a sequence of operations on the two stacks. Your implementation should require $O(1)$ amortized time per operation. Prove that indeed your data structure achieves this amortized time bound.

2. We maintain items in an array subject to the following operations.

   1. insert(i): insert item i in the next free slot of the array.
   2. delete: delete an item from the last occupied slot of the array.

Describe an implementation for these operations such that each operation takes $O(1)$ amortized time and the length of the array at any time is proportional to the number of occupied slots. (I.e. there should be constants c_1 and c_2 such that $c_1 n \leq L \leq c_2 n$ at any time where L is the length of the array and n is the number of occupied slots.) Prove that your implementation indeed achieves these time bounds.

3. 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.

4. 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.

5. AVL-trees (or height balanced trees) are another popular family of balanced binary trees. A binary search tree is an AVL tree if for each internal node u with children v_1 and v_2

$$|height(v_1) - height(v_2)| \leq 1$$

where the height of a node is the longest path from that node to a leaf. We maintain with each node the difference between the height of its left child and the height of its right child.

Suggest implementations of insert and delete for AVL trees that take $O(\log n)$ worst-case time.

Does your implementation support a sequence of m insertions in $O(m)$ time ? If so prove that it does and if not provide a counterexample.

Does your implementation support a sequence of m insertions and deletions in $O(m)$ time ? If so prove that it does and if not provide a counterexample.

6. Complete the proof of correctness of the algorithm by Garsia and Wachs presented in class.