TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Spring Semester, 2002/2003

**Homework 1, March 3, 2003**

**Due on March 17. 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 two 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. 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.

5. Consider the *dynamic* inverse range searching problem (point-enclosure) defined in class. Suppose that for any two segements $s_1$ and $s_2$ that are together in the data structure (at any time) either $s_1 \subseteq s_2$, $s_2 \subseteq s_1$, or $s_1 \cap s_2 = \emptyset$. Describe a solution that exploits this restriction and analyse it.

6. You are given a set $S$ of segments on the line where each $s \in S$ has a priority $p(s)$ associated with it. Describe how to represent $S$ in a data structure such that given a query point $x$ we can efficiently find a segment of highest priority containing $x$. What is the preprocessing time of your solution ? How much space does it need ? and how fast can it answer a query ?

b) Describe a data structure that in addition to queries as defined above can also support insertions of new segments (with their associated priorities.) Analyse the preformance of your solution.

c) Describe a data structure that can support queries, insertions of new segments, and deletions of segments. Analyse your data structure.