Second Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is October 02.

Problem 1. ($20 = 12 + 8$ points). Consider an array $A[1..n]$ for which we know that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that i is a *local minimum* if $A[i-1] \geq A[i] \leq A[i+1]$. Note that A has at least one local minimum.

(a) We can obviously find a local minimum in time $O(n)$. Describe a more efficient algorithm that does the same.

(b) Analyze your algorithm.

Problem 2. (20 points). A *vertex cover* for a tree is a subset V of its vertices such that each edge has at least one endpoint in V. It is *minimum* if there is no other vertex cover with a smaller number of vertices. Given a tree with n vertices, describe an $O(n)$-time algorithm for finding a minimum vertex cover. (Hint: use dynamic programming or the greedy method.)

Problem 3. (20 points). Consider a red-black tree formed by the sequential insertion of $n > 1$ items. Argue that the resulting tree has at least one red edge.

[Notice that we are talking about a red-black tree formed by insertions. Without this assumption, the tree could of course consist of black edges only.]

Problem 4. (20 points). Prove that $2n$ rotations suffice to transform any binary search tree into any other binary search tree storing the same n items.

Problem 5. ($20 = 5 + 5 + 5 + 5$ points). Consider a collection of items, each consisting of a key and a cost. The keys come from a totally ordered universe and the costs are real numbers. Show how to maintain a collection of items under the following operations:

(a) $\text{ADD}(k, c)$: assuming no item in the collection has key k yet, add an item with key k and cost c to the collection;

(b) $\text{REMOVE}(k)$: remove the item with key k from the collection;

(c) $\text{MAX}(k_1, k_2)$: assuming $k_1 \leq k_2$, report the maximum cost among all items with keys $k \in [k_1, k_2]$.

(d) $\text{COUNT}(c_1, c_2)$: assuming $c_1 \leq c_2$, report the number of items with cost $c \in [c_1, c_2]$;

Each operation should take at most $O(\log n)$ time in the worst case, where n is the number of items in the collection when the operation is performed.