Third Homework Assignment

Write the solution to each problem on a single page. The deadline for handing in solutions is October 14.

Problem 1. ($20 = 10 + 10$ points). Consider a lazy version of heapsort in which each item in the heap is either smaller than or equal to every other item in its subtree, or the item is identified as *uncertified*. To *certify* an item, we certify its children and then exchange it with the smaller child provided it is smaller than the item itself. Suppose $A[1..n]$ is a lazy heap with all items uncertified.

(a) How much time does it take to certify $A[1]$?

(b) Does certifying $A[1]$ turn A into a proper heap in which every item satisfies the heap property? (Justify your answer.)

Problem 2. (20 points). Recall that Fibonacci numbers are defined recursively as $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. Prove the square of the n-th Fibonacci number differs from the product of the two adjacent numbers by one: $F_n^2 = F_{n-1} \cdot F_{n+1} + (-1)^{n+1}$.

Problem 3. (20 points). Professor Pinocchio claims that the height of an n-node Fibonacci heap is at most some constant times $\log_2 n$. Show that the Professor is mistaken by exhibiting, for any integer n, a sequence of operations that create a Fibonacci heap consisting of just one tree that is a linear chain of n nodes.

Problem 4. ($20 = 10 + 10$ points). To search in a sorted array takes time logarithmic in the size of the array, but to insert a new items takes linear time. We can improve the running time for insertions by storing the items in several instead of just one sorted arrays. Let n be the number of items, let $k = \lceil \log_2(n + 1) \rceil$, and write $n = n_{k-1} n_{k-2} \ldots n_0$ in binary notation. We use k sorted arrays A_i (some possibly empty), where A_i stores $n_i 2^i$ items. Each item is stored exactly once, and the total size of the arrays is indeed $\sum_{i=0}^{k} n_i 2^i = n$. Although each individual array is sorted, there is no particular relationship between the items in different arrays.

(a) Explain how to search in this data structure and analyze your algorithm.

(b) Explain how to insert a new item into the data structure and analyze your algorithm, both in worst-case and in amortized time.

Problem 5. ($20 = 10 + 10$ points). Consider a full binary tree with n leaves. The *size* of a node, $s(\nu)$, is the number of leaves in its subtree and the *rank* is the floor of the binary logarithm of the size, $r(\nu) = \lfloor \log_2 s(\nu) \rfloor$.

(a) Is it true that every internal node ν has a child whose rank is strictly less than the rank of ν?

(b) Prove that there exists a leaf whose depth (length of path to the root) is at most $\log_2 n$.