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

Homework 2, March 24, 2003

Due on April 7, please keep a copy of your homework before handing it in.

1. For any positive integer n, give a sequence of Fibonacci heaps operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes (make your sequence as short as you can).

2. Consider an implementation of Fibonacci heaps without cascading cuts (all other details are as shown in class, the only difference is that delete and decrease-key just cut the subtree and do not continue with cascading cuts). For any large enough m show a sequence of m operations on heaps of size at most n such that the average cost of an operation is as high as possible. (By m large enough we mean larger even than some function of n.)

3. Given a binary tree T, attach a "dummy" leaf where there is a null pointer. We refer to these "dummy" leaves as *external nodes*. In a binary tree T define the rank of a node to be the minimum length (in edges) of a path from x to an external node. A *leftist* tree is a binary tree in which for every node x, $rank(left(x)) \geq rank(right(x))$, where $left(x)$ is the left child of x and $right(x)$ is the right child of x.
a) Prove that the rightmost path in a leftist tree is a shortest path from the root to a leaf and that its length is at most $\log_2(n+1)$.

A *leftist* heap is a leftist tree containing one item per internal node in heap order.

b) Describe an efficient implementation of *meld* operation on leftist heaps. What is the running time of your implementation ?
c) Using your implementation of *meld* describe an efficient implementation of *insert* and *deletemin*. What is the running time of your implementations ?
d) Design an implementation of the operation $heapify(q)$. This operation takes a list q of k leftist heaps containing n items alltogether and melds them into a single heap. What is the running time of your implementation ?
e) How would you implement the round-robin algorithm of Cheriton and Tarjan for finding a minimum spanning tree using leftist heaps ? what is the running time of your implementation ?

4. Describe a heap data structure that supports find-min in $O(1)$ time, insert and meld in $o(\log(n))$ time (recall that little "o" means asymptotically faster than $\log n$), and as fast Delete-min as you can. All time bounds are on the worst case. You'd get extra credit if you do meld and insert in $O(1)$ time and delete-min in $O(\log n)$.

5. Augment the redundant binary counters shown in class to support both increment and decrement in $O(1)$ worst case time. Describe the algorithm that decrements the counter and establish its correctness.