Problem Set 2

Due: Monday, September 15, 2003.

Problem 1 is marked **noncollaborative**. You may not discuss it or related material with anyone else.

Problem 5 is marked **optional**. If you solve it, it will be graded, but you will not receive extra credit for it. It will not affect your grade.

NONCOLLABORATIVE Problem 1. Given two strings x and y, show how to find the longest common substring z, i.e., the longest string z that is a substring of both x and y. The running time of your algorithm should be linear, i.e., $O(|x|+|y|)$. Hint: Use suffix trees.

Problem 2. Professor Pinocchio claims that the height of an n-node Fibonacci heap is $O(\log n)$. Show that the professor is mistaken by exhibiting, for any positive integer n, a sequence of Fibonacci heap operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes.

Problem 3. More on amortizing heap operations. The bounds asked for can of course be derived from Fibonacci heaps; the goal is to work with the data structures specified rather than introducing complicated new ones.

(a) Let P be a priority queue that performs insert, delete-min, and merge in $O(\log n)$ time, and performs make-heap in $O(n)$ time where n is the size of resulting priority queue. Show that P can be modified to perform insert in $O(1)$ amortized time, without affecting the cost of delete-min or merge (i.e. $O(\log n)$ amortized time). Assume that the priority queue does not support an efficient decrease-key operation.

(b) Using the above technique, show that even binary heaps can be modified to support insert in $O(1)$ amortized time while maintaining an $O(\log n)$ time bound for delete-min. Note that binary heaps do not support merge in $O(\log n)$ time.

Problem 4. In class we saw how to use a van Emde Boas priority queue to get $O(\log\log u)$ time per queue operation (insert, delete-min, decrease-key) when the range of values is $\{1, 2, \ldots, u\}$. Show that for the single-source shortest paths problem on a graph with n nodes and range of edge lengths $\{1, 2, \ldots, C\}$, we can obtain $O(\log\log C)$ time per queue operation, even though the range of values is $\{1, 2, \ldots, nC\}$,

OPTIONAL Problem 5. Keeping a mark bit around in Fibonacci heaps may be wasteful. Suppose that instead, each time I do a cut, I flip a coin to decide whether to cascade that cut to the parent. If the coin is unbiased, show that the expected behavior of these markless Fibonacci heaps is like that of standard ones. Can I bias the coin, so that a cascade is **more** than 50% likely, to achieve the effect of cascading after (say) one and a half children are cut? Can this be used to improve the time for delete-min at the cost of increasing the time for decrease key?