CS 423: Theory of Algorithms
Spring, 1999
Tarjan

Problem Set 3
Due: Thursday, March 4
Half credit if turned on Tuesday, March 9

Problem Set No. 3

1. (Alternative solution to problem 2 on set 2) Prove that an intermixed sequence of $m \geq n$ finds and some number of intermixed links on an initial collection of n singleton sets takes $O(m \log n)$ time, using the following potential function. Define the *size* $s(x)$ of a node x to be the total number of its descendants, including x itself. Observe that $s(x)$ can only increase because of a link, and can only decrease because of a find. Define $\phi(x) = \log_2 s(x)$, and define Φ to be the sum of all the node potentials.

2. (sums over tree paths) Consider a collection of n single node rooted trees, with each node x having an associated real number *value* (x). Consider two kinds of operations:

   graft (x, y)**:** Make node y the parent of node x. This operation is only allowed if x is a tree root before the graft and y is a node in a different tree before the graft; y *need not* be a root.

   sum (x)**:** Compute and return the sum of the values of all ancestors of x, including x itself.

(a) Show how to implement these operations so that $m \geq n$ sums intermixed with grafts take $O(m \log n)$ time.
Hint: Use path compression with naive linking.

(b) Improve the running time to $O(m\alpha(n))$.
Hint: Use path compression with linking by rank.

(This problem generalizes problem 22-2 in CLR.)

3. Show by example that a decrease-key operation on an n-item Fibonacci heap can take $\Omega(n)$ time.

4. (Finger Search Trees)

   In a binary tree, the *left spine* is the path from the root through left children to the smallest node in symmetric order; the *right spine* is defined symmetrically. Consider a modified red-black tree in which each node on the left spine points to its parent and its right child (instead of its left and right children) and each node on the right spine points to its left child and parent (instead of its left and right children). Access is by two pointers, to the first and last nodes in symmetric order. We call such a tree an *r-b finger search tree.*

(a) Describe a search algorithm on r-b finger search trees such that accessing the k^{th} smallest or k^{th} largest item takes $O(\log k)$ time.

(b) Describe insertion and deletion algorithms on r-b finger search trees such that inserting a new k^{th} smallest or k^{th} largest item takes $O(\log k)$ amortized time.

Hint: For (b), a suitable potential function is to assign two units of potential to a black node with two red children, one unit of potential to a black node with two black children, and zero potential to other nodes. Assume that the initial search tree is empty.