Due: Friday, April 24, 2009 at the beginning of class.

Problem 1. Leftist & Skew Heap Insert

In this problem, we will build leftist and skew heaps by inserting values one at a time.

(a) Show the result of inserting keys 1 to 7 in ascending order (i.e. insert 1, then 2, then 3, etc.) into an initially empty leftist heap. Use the leftist heap insert (i.e. merge) algorithm at each step. Again, intermediate steps are not required, but may help you earn partial credit.

(b) Show the result of inserting keys 1 to 7 in ascending order (i.e. insert 1, then 2, then 3, etc.) into an initially empty skew heap. Use the skew heap insert (i.e. merge) algorithm at each step. Again, intermediate steps are not required, but may help you earn partial credit.

(c) Prove or disprove: A perfect binary tree forms if keys $1$ to $2^k - 1$ are inserted in order (again this means 1 first, then 2 etc) into an initially empty skew heap. $k$ is a positive integer. (Hint: use induction on $k$.)

Problem 2. Merging Skew Heaps

Weiss 6.26. You only need to show the final result, but note that if you do this it will be hard to award partial credit if the final result has problems.

Problem 3. Binomial Trees and Queues

A binomial tree of height 0, $B_0$, is a one-node tree. A binomial tree of height $k$, $B_k$ is formed by attaching a binomial tree, $B_{k-1}$ to the root of another binomial tree another binomial tree $B_{k-1}$. (These are the same definitions as in Weiss.)

(a) Weiss 6.32.

(b) Prove that a binomial tree $B_k$ has $2^k$ nodes.

Problem 4. AVL Trees

Show the result of inserting 8, 9, 5, 4, 1, 7, 3, then 2 into an initially empty AVL tree.

Problem 5. Verifying AVL Trees

Weiss 4.22. You must verify that the balance condition is correct, and also that the height value has been correctly maintained. In your answer, assume a tree structure like the following:

class Node {
    int data;
    int height;
    Node left, right;
}