CSE 326 DATA STRUCTURES
HOMEWORK 3

Due: **Friday, January 29, 2010** at the beginning of class. Please put your quiz section (AA, AB, AC) in addition to your name at the top of your homework.

Problem 1. Leftist Heaps

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

(a) Weiss 6.19. 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.

(b) Show the result of inserting keys 0 to 7 in ascending order (i.e. insert 0 , then 1, then 2, 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.

Problem 2. Skew Heaps

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

(a) 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.

(b) Show the result of inserting keys 0 to 7 in ascending order (i.e. insert 0 , then 1, then 2, 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 $2^k - 1$ keys are inserted in ascending order into an initially empty skew heap. k is a positive integer. (Hint: use induction on k.)

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. 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.

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

Problem 4. increaseKey and decreaseKey for Skew Heaps and Binomial Queues

Recall that, given a pointer to an object in a binary heap, we can perform the decreaseKey operation in the object by decreasing its key value and then performing a percolateUp operation. Similarly, we can perform the increaseKey operation by increasing the key value and then doing a percolateDown. For binary heaps of size

n, percolateUp and percolateDown are both $O(\log n)$ in the worst case; thus decreaseKey and increaseKey are $O(\log n)$ in the worst case for binary heaps.

Here we ask you to analyze the performance of decreaseKey and increaseKey for skew heaps and binomial queues. For this problem, you will assume the following: decreaseKey and increaseKey are again implemented by adjusting the key value of a node and performing either a percolateUp or a percolateDown operation to satisfy the heap order property. As above, the pointer to the object (i.e. the node) is given. The parent of a node can be discovered in constant time (e.g. with an extra pointer per node) In the analysis below, "complexity" refers to the big-o bound. Given these assumptions, answer the following questions. Note: The justification for each of your answers is a large part of the credit to be earned.

(a) What is the worst case complexity of performing a decreaseKey operation on a skew heap using percolateUp? Justify your answer.

(b) What is the worst case complexity of performing a increaseKey operation on a skew heap using percolateDown? Justify your answer.

(c) What is the worst case complexity of performing a decreaseKey operation on a binomial queue using percolateUp? Justify your answer.

(d) What is the worst case complexity of performing a increaseKey operation on a binomial queue using percolateDown? Justify your answer.