CSE332 Data Abstractions, Summer 2010

Homework 2

Due Friday, July 9, 2010 at the beginning of class.

Problem 1: Binary Heaps

This problem gives you some practice with the basic operations on binary min heaps. Make sure to check your work.

a. Starting with an empty binary min heap, show the result of inserting, in the following order, 12, 9, 3, 8, 5, 6, 14, 1, 7, 11 and 2, one at a time (using percolate up each time), into the heap. By show, we mean, "draw the tree resulting from each insert."

b. Now perform two deleteMin operations on the binary min heap you constructed in part (a). Show the binary min heaps that result from these successive deletions, again by drawing the binary tree resulting from each delete.

c. Instead of inserting the elements in part (a) into the heap one at a time, suppose that you use the linear-time buildHeap operation known as Floyd's algorithm. Show the binary min heap tree that results from buildHeap. (It will help to show some intermediate trees so that if there are any bugs in your solution we will be better able to assign partial credit, but this is not required.)

Problem 2: Merging Perfect Trees

If we have two binary min heaps h1 and h2 of total size n, then Floyd's algorithm lets us combine them to build a new heap with all the elements from the two heaps in time O(n). However, there are special cases where we can do better. Here we consider the case where h1 and h2 are both perfect trees: a perfect tree is a binary tree where every level i contains 2^i nodes. That is, the rows of h1 and h2 are always full. We also allow the merge operation to "re-use/destroy" h1 and h2, i.e., only the merged heap is available after the merge operation. We assume the heaps are represented as pointer-based trees (not arrays) but we can find the "last" element of h1 and h2 in O(1) time.

a. Suppose h1 and h2 have the same height. Describe an O(log n) algorithm to merge the heaps. A concise English answer is sufficient.

b. Suppose the height of h1 is the height of h2 minus one. Describe an O(log n) algorithm to merge the heaps. A concise English answer is sufficient.

Problem 3: Alternative Remove

One way to remove an arbitrary object from a binary min heap is to decrease its priority value by infinity and then call deleteMin. An alternative is to remove it from the heap, thus creating a hole, and then repair the heap.

a. Write pseudocode for an algorithm that performs the remove operation using the alternative approach described above. Your pseudocode should implement the method call remove(int index), where index is the index into the heap array for the object to be removed – assume an array representation. Your pseudocode can call the following methods described in lecture: insert, deleteMin, percolateUp, and percolateDown. Like in lecture, you may assume that objects are just priority integers (no other data).

b. What is the worst case complexity of the algorithm you wrote in part (a)?

Problem 4: AVL Insertion

Show the result of inserting 12, 8, 4, 9, 5, 6, 11, 2, 7 and 1 in that order into an initially empty AVL tree. Show the tree after each insertion, clearly labeling which tree is which.

Problem 5: Verifying AVL Trees

Give pseudocode for a linear-time algorithm that verifies that an AVL tree is correctly maintained. Assume every node has fields key, data, height, left, and right and that keys can be compared with <, ==, and >. The algorithm should verify all of the following:

- The tree is a binary search tree.
- The height information stored in each node is correct.
- Every node is balanced.

Your code should have a Boolean return type, and return true if the tree is a valid AVL tree, or false if it violates any of the above properties.