EECS 311: Data Structures and Data Management Homework 3
Assigned: 10/10/08 Due: 10/17/08

1 Reading.

Chapter 4, Sections 1-4.

2 Problems.

1. Problem 4.6.
2. Problem 4.8.
3. Give an algorithm for *find-min* that returns the smallest key in a binary search tree. What is the worst-case runtime of your algorithm on an n-node tree of height h? What is the worst-case runtime of your algorithm on any n-node binary search tree? What is the worst-case runtime of your algorithm on a *balanced* binary search tree?
4. Consider a dictionary ADT that supports *find-min* in addition to the standard dictionary ADT operations of *insert*, *remove*, *find*, and *is-empty*. Use the dictionary to implement a sort routine. You may assume that the sort routine is given its input in a length n array and is to place its output on a length n array.
5. Consider the sort routine from Problem 4.
   (a) Assuming all dictionary operations have $O(\log n)$ worst-case runtime, what is the worst-case runtime of your sort algorithm?
   (b) Suppose that dictionary operations are only amortized $O(\log n)$ each; i.e., starting from creation any sequence of m operations takes at most $O(m \log n)$, where n is the maximum size of the dictionary. What is the worst-case runtime of your sort algorithm (write your bound in terms of n only)?
   (c) In terms of our sort routine, discuss the pros and cons, if any, of worst-case $O(\log n)$ dictionary operations verses the amortized $O(\log n)$ dictionary operations.