

1 Reading.

Chapter 4, Sections 1-4.

2 Problems.

1. Calculate the minimum and maximum number of nodes in a binary tree of height h . Explain the scenarios that give these minimum and maximum numbers.
2. Suppose a binary tree has leaves that are numbers and internal nodes that are binary arithmetic operators. An postorder traversal of one such tree prints out “2 3 * 4 2 * 6 - +”.
 - (a) Draw the tree with the above postorder traversal.
 - (b) Give the preorder and inorder traversal of this tree.
 - (c) Evaluate the numeric calculation suggested by the tree.
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*. Give an algorithm for sorting tht uses this dictionary. 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.