
Homework 7 – Due Friday, March 30, 2007

Reminder

- Collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.

Exercises These should not be handed in, but the material they cover may appear on exams.

- Non-starred exercises in Chapter 14 (replace “red-black tree” with “binary search tree”).

Problems to be handed in

1. (**2-3 trees**) Read over the lecture notes on the course homepage about 2-3 trees.
 - (a) (**Practice/review**) Consider the 2-3 tree on slide 72 (page 18) of the notes. Draw the tree resulting from performing each of the following operations, in order:
 - i. Delete 37
 - ii. Insert 21
 - iii. Delete 70
 - iv. Insert 41
 - (b) (**Predecessor**) Give an algorithm for the PREDECESSOR operation in a 2-3 tree (we saw a similar algorithm in class for binary search trees, see Chapter 12.2). Give *both* a concise English description *and* pseudocode.
 - (c) (**Traversals**) Using a postorder traversal of a 2-3 tree, give an algorithm which computes the smallest gap between any two keys in the tree. Your algorithm should run in time $\Theta(n)$. Give *both* a concise English description *and* pseudocode.
2. (**Augmented Data Structures**, similar to exercise 14.2-1)

Augment a binary search tree so that the MAXIMUM, MINIMUM, PREDECESSOR and SUCCESSOR operations all can be implemented in constant time (N.B.: recall that all these operations take a pointer to a node as input and return a pointer to a node, not just a key). Explain why the insert and delete operations still take time $O(h)$ where h is the height of the tree. It may help to look at the proof of Theorem 14.1 in the CLRS book.

Organize your answer using the methodology seen in class (CLRS, Section 14.2).

3. (**Range Search**, similar to Exercise 14.2-5) We return now to vanilla-model, nonaugmented search trees.
- (a) In a regular, nonaugmented search tree, the operation $\text{RANGE-SEARCH}(x, a, b)$ outputs all the keys k such that $a \leq k \leq b$ in the tree rooted at x .
Give a simple recursive algorithm for RANGE-SEARCH
- in a binary search tree
 - in a 2-3 tree
- Give both an English description and pseudocode.
- (b) State and prove the running time of your BST algorithm (part i) in terms of m and h , where m is the number of keys that match the initial query and h is the height of the tree.
To get full credit, your algorithm should run in time $\Theta(m+h)$ and you must give a proof of its running time.
- (c) Consider the augmented data structure of Problem 2. Can you use the new operations to get a RANGE-SEARCH algorithm which is simpler to analyze and still runs in time $\Theta(m+h)$?
4. (**Extra credit**) Solve exercise 14.3-7.