CSPP 55001 Algorithms — Autumn 2009

Homework 5 (assigned October 28, due November 4)

Reading: CLRS chapters 9, 11, and 12.

Written assignment: Solve the following "Do" exercises and assigned problems. **Only solutions to the assigned problems should be turned in.**
Note: You are responsible for the material covered in **both** "Do" exercises and assigned problems.
Note: If you work with others, indicate their names at the top of your homework paper. Everyone must submit their own independently written solutions.

"Do" Exercises (*not* to be handed in):

1. Exercise 9.3-1 on page 223.
   2nd Edition, Exercise 9.3-1, page 192.
2. Problem 9-2, parts a–c, on page 225.
   2nd Edition, Problem 9-2, parts a–c, page 194. Note that the x_i are not necessarily given in sorted order.
3. Exercise 11.4-1 on page 277.
   2nd Edition, Exercise 11.4-1, page 244.
4. Problem 11-4, parts a–b, on page 284.
   2nd Edition, Problem 11-4, parts a–b, page 251.
5. Exercises 12.3-4, 12.3-5 on page 299.
   2nd Edition, Exercises 12.3-4, 12.3-5 on page 264.

Problems (to be handed in):

1. You are given two **sorted** arrays A and B, each containing n numbers. Give an $O(\lg n)$-time algorithm to find the median of all $2n$ numbers. Describe your algorithm in pseudocode. Argue (informally) that your algorithm is correct and analyze its running time. (15 points).

2. You are given a set S of n distinct numbers and a positive integer $k \leq n$. Give an $O(n)$ worst-case-time algorithm that determines the k numbers in S that are closest to the median of S. Argue (informally) that your algorithm is correct and analyze its running time. Note: The $O(n)$ bound does not depend on k. (15 points)

3. Problem 11-1, parts a–b, on page 282. (5 points each)
   2nd Edition, Problem 11-1, parts a–b, pages 249–251.

4. (1) Describe an efficient algorithm to merge two balanced binary search trees with n elements each into a balanced binary search tree. Analyze the running time of your algorithm. (10 points)
   (2) Two binary search trees T_1 and T_2 are said to be *equivalent* if they contain exactly the same elements. That is, for all x in T_1, x in T_2, and for all y in T_2, y in T_1. Describe an efficient algorithm to determine if two BSTs T_1 and T_2 are equivalent. Assume that each BST has n elements. Analyze the running time of your algorithm. (10 points)
   (3) Prove that if we start with a node that has k successors in a height-h binary search tree, k

successive calls to the procedure **Tree-Successor** take $O(k+h)$ time. (See page 292 (2nd ed., page 259) for **Tree-Successor** procedure.) (10 points)

Gerry Brady
Thursday October 29 17:42:09 CDT 2009