CISC 320 010 Introduction to Algorithms (Fall 2005)

Homework 2
Handed out: September 20, 2005
Due date: October 6, 2005

Your handwriting must be legible, and your answers should be rigorous, concise and in proper order. Please note that the work handed in must be your own.

1. (35 points) Sort the array *A = <3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7>* using the Quciksort algorithm as given in the text (CLRS).

(a) Show what the array looks like after the PARTITION is called once. (Use Figure 7.1 as a model, but you only need to show the array when PARTITION is returned).

(b) Tell whether the relative positions of any identical keys – e.g., there are three 5's – are changed, after the PARTITION. You may want to mark the identical keys with superscripts to differentiate them, e.g., 3^a, 1^a, 4, 1^b, 5^a, 9^a, 2, 6, 5^b, 3^b, 5^c, 8, 9^b, 7.

(c) Draw the recursion tree. In the tree, each node, corresponding to a call to the PARTITION, has two boxes: one is filled with the input size for partitioning, and the other is filled with number of comparisons done by the PARTITION.

(d) How many times is the PARTITION called?

(e) What is the depth of the recursion tree?

(f) How many comparisons are made?

2. (25 points) Sort the array *A = <5, 3, 17, 10, 84, 19, 6, 22, 9>* using the Heapsort algorithm as given in the text (CLRS).

(a) Show the array after the BUILD-MAX-HEAP is returned.
(b) How many comparisons are made by the BUILD-MAX-HEAP?
(c) How many comparisons are made in total after the array is sorted?
(d) Is an array in increasing order a best case, worst case, or intermediate case for the Heapsort? Justify your answer.

3. (20 points) (CLRS 9-1) Given a set of *n* numbers, we wish to find the *i* largest in sorted order using a comparison-based algorithm. Analyze the running times of the following three algorithms in terms of *n* and *i*, and indicate which algorithm gives the best asymptotic worst-case running time.

a. Sort the numbers, and list the *i* largest.
b. Build a max-priority queue from the numbers, and call EXTRACT-MAX *i* times.
c. Use an order-statistic algorithm to find the *i*-th largest number, partition around that number, and sort the *i* largest numbers.

4. (20 points) Describe a $\Theta(n \log n)$ – time algorithm that, given an unordered set *S* of *n* integers and another integer *x*, determines whether there exist two integers in *S* whose sum is exactly *x*. Write pseudocode.