Ph.D. Comprehensive Exam:
Math and Algorithms

Fall 2002

The exam includes eight problems;
the length of the exam is three hours.

Problem 1

The double factorial, $n!!$, is defined by the following recurrence:

> $0!! = 1!! = 1$;
> for $n \geq 2$, $n!! = n \cdot (n-2)!!$.

For example, $6!! = 2 \cdot 4 \cdot 6 = 48$, and $7!! = 1 \cdot 3 \cdot 5 \cdot 7 = 105$.
Prove or disprove the following asymptotic bound:

$$n!! = o((n+1)!!).$$

Problem 2

Determine asymptotically tight bounds (Θ-notation) for the following recurrences, and show the derivation of your bounds:

(a) $T(n) = 3 \cdot T(n/2) + 4 \cdot T(n/4) + n^2$.

(b) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$.

Problem 3

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, the pair (i, j) is called an inversion. For example, the array $\langle 2, 3, 8, 6, 1 \rangle$ has five inversions. Give an algorithm that determines the number of inversions in $A[1..n]$; its running time must be $O(n \cdot \lg n)$.

Problem 4

Give a *linear-time* algorithm that converts a sorted array $A[1..n]$ into a *balanced* binary search tree. That is, the algorithm should input $A[1..n]$ and construct an n-node balanced tree that includes all elements of the array.

Problem 5

Give a linear-time *nonrecursive* algorithm that outputs all elements of a binary search tree in sorted order.

Problem 6

Suppose that we augment a normal programming language with an additional “magic” function, Magic-Max(A, i, j). The arguments of this function include an array $A[1..n]$ and two indices, i and j, such that $1 \leq i \leq j \leq n$. The function sometimes returns the index of the largest element in $A[i..j]$, and sometimes the index of the second largest element in $A[i..j]$; its choice between the largest and second largest element is random. The magic property of this function is its speed; specifically, it returns an answer in *constant time.* Your task is to use this language to develop a procedure that sorts an array of real values in *linear time.* It must always return the correct sorting, and its worst-case time must be linear.

Problem 7

Suppose that S is a finite set of *natural* numbers, and we need to determine whether there is a subset S' of S whose elements sum to 2002. That is, we have to construct an algorithm that inputs S, and returns TRUE if there exists $S' \subseteq S$ such that $\sum_{x \in S'} x = 2002$. Determine whether this problem is NP-hard and justify your answer.

Problem 8

We define the length of a path in an unweighted graph as the number of edges in the path. We consider the task of finding a longest *simple* path between two given vertices in an undirected unweighted graph; recall that a path is simple if it has no self-intersections. Determine whether this problem is NP-hard and justify your answer.