Illinois Institute of Technology
Department of Computer Science

First Examination

CS 430 Introduction to Algorithms
Spring, 2012

11:25am–12:40pm, Wednesday, February 8, 2012
104 Stuart Building

Print your name and student ID, *neatly* in the space provided below; print your name at the upper right corner of *every* page. Please print legibly.

Name:	
Student ID:	

This is an *open book* exam. You are permitted to use the textbook, any class handouts, anything posted on the web page, any of your own assignments, and anything in your own handwriting. Foreign students may use a dictionary. *Nothing else is permitted*: No calculators, laptops, cell phones, Ipods, Ipads, etc.!

Do all five problems in this booklet. *All problems are equally weighted, so do not spend too much time on any one question.*

Show your work! You will not get partial credit if the grader cannot figure out how you arrived at your answer.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Randomized Max**

   Consider the following algorithm to find the maximum element in an unordered array of n elements:

```
1: RandomMax($A[1..n]$)
2: $m \leftarrow -\infty$
3: for $i \leftarrow 1$ to $n$ in random order do
4:   if $A[i] > m$ then
5:     $m \leftarrow A[i]$
6:   end if
7: end for
8: return $m$
```

   Notice that the **for** loop (line 3) takes the numbers $1, 2, \ldots, n$ in random order.

   (a) In the worst case, how many times does RandomMax execute line 5?

   (b) What is the probability that nth (last) iteration executes line 5?

   (c) Analyze the expected number of executions of line 5.

2. **Majority Rules**

   Given a set $\{x_1, x_2, \ldots, x_n\}$, each element of which is colored either red or blue, design an algorithm to find the majority in a set of items by making equal/not equal color comparisons $x_u : x_v$; when n is even, you must report that there is no majority if there are equal numbers of each color. Analyze the performance of your algorithm. For full credit, your algorithm should take time $O(n)$.

   (*Hint*: If there are two elements $x_i \neq x_j$ in the set, and a new set is formed by eliminating these two elements, then the majority in the the original set remains a majority in the new set.)

3. **Lucky Quicksort**

   Suppose you are very lucky and your implementation of quicksort always picks a partitioning element between the 25th and 75th percentiles of the (sorted) values. Analyze the worst-case performance of this version of quicksort.

4. **Binary Search Trees**

   Argue that since sorting n elements takes $\Omega(n \log n)$ time in the worst case in the comparison model, any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \log n)$ time in the worst case.

5. **Red-Black Trees**

   What is the largest possible number of internal nodes in a red-black tree with black-height k? What is the smallest possible number? What if we allow the root to be red?