Illinois Institute of Technology
Department of Computer Science

First Examination

CS 430 Introduction to Algorithms
Spring, 2009

11:25am–12:40pm, Wednesday, February 23, 2009
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, 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 Min**

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

```
1: RandomMin(A[1..n])
2: m ← ∞
3: for i ← 1 to n in random order do
4:    if A[i] < m then
5:       m ← 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 RandomMin 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. **Unusual Sorting Algorithm**

A TA in CS 430 suggested the following unusual algorithm to sort n elements in an array:

```
1: TaSort(A[0..n − 1])
2: if n = 2 and A[0] > A[1] then
3:    swap A[0] ↔ A[1]
4: else
5:    if n > 2 then
6:       m ← ⌈2n/3⌉
7:       TaSort(A[0..m − 1])
8:       TaSort(A[n − m..n − 1])
9:       TaSort(A[0..m − 1])
10:   end if
11: end if
```

(a) Prove that TaSort correctly sorts its input.

(b) When the TA first proposed the algorithm, she mistyped line 6 using the floor instead of the ceiling operation. The algorithm failed to sort correctly; explain why.
(*Hint*: Consider $n = 4$.)

(c) State (using precise floor/ceiling operations) the recurrence relation and initial values describing the number of comparisons $A[0] > A[1]$ performed in line 2 of the algorithm in the worst case. Then, ignoring the floor/ceiling operations, solve the recurrence.

3. **Special Hardware Priority Queue**

Distressed by the results of the previous problem, the TA has developed a hardware priority queue for his computer. The priority queue device can store up to p records, each consisting of a key and a small amount of satellite data (such as a pointer). The computer to which it is attached can perform INSERT and EXTRACTMIN operations on the priority queue, each of which takes $O(1)$ time, no matter how many records are stored in the device. The TA wishes to use the hardware priority queue to help implement a sorting algorithm on his computer. He has n records stored in the primary memory of his machine. If $n \leq p$, the TA can certainly sort the keys in $O(n)$ time by first inserting them into the priority queue, and then repeatedly extracting the minimum. Design an efficient algorithm for sorting $n > p$ items using the hardware priority queue. Analyze your algorithm in terms of both n and p.

4. **Augmented Binary Search Trees**

   Suppose we augment a red-black tree so each node has a field giving the height of its subtree. Show that a rotation in such a tree can be done in time $O(1)$, correctly maintaining the height fields.

5. **Maximum External Path Length in Binary Search Trees**

   Given a binary tree with n internal nodes and $n + 1$ leaves, what is the maximum possible external path length? Prove your answer.