Illinois Institute of Technology
Department of Computer Science

Second Examination

CS 430 Introduction to Algorithms
Fall, 2014

Wednesday, October 29, 2014
10am–11:15pm, 121 Life Sciences
11:25am–12:40pm, 113 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. **Augmenting Red-Black Trees**

   Consider a binary tree in which every internal node x has an associated frequency $f(x)$ which is stored in the node. Let $F(x)$ be the frequency of occurrence of *all nodes* in the subtree rooted at x; leaves have frequency 0. Can red-black trees be augmented with the $F(x)$ values without affecting the $O(\log n)$ performance of the insertion and deletion algorithms? Prove your answer.

2. **Dynamic Programming**

Money is spread over the squares of an $n \times n$ checkerboard; square i, j has $c_{i,j}$ dollars on it. You start at square 1,1 (the upper left square of the board) and can move one step down or right at each move (never up or left), collecting the money on the squares you cross as you move to the bottom right square n, n.

(a) Let $C[i, j]$ be the largest amount that you can accumulate when you reach square i, j. Give a recurrence relation for $C[i, j]$, including initial conditions, based on the Principle of Optimality.

(b) Give a memoized, iterative algorithm based on (a), and analyze it.

(c) What additional memoization is needed to determine the *sequence of steps* used in accumulating the largest amount of money?

(d) **Extra credit** Analyze a recursive, *unmemoized* algorithm based on your recurrence in part (a); you need not write the algorithm.

2. **Dynamic Programming, continued.**

3. **Greedy Heuristics**

Professor Reingold is planning to have n problems on Exam 3. There happen to be exactly n students in the course and each has discovered a different one of the exam problems. The students want to share their information by sending email, so that every student knows every problem. Assume that a student includes all the problems she/he knows at the time a message is sent and that email can go only to one recipient.

(a) Give a greedy algorithm organizing the email communication, trying to minimize the total number of email messages for the students to fully share their knowledge.

(b) Prove that your greedy algorithm results in the fewest messages possible, or give an example where it does not.

4. **Amortized Analysis**

We saw in class (October 15) and in CLRS3 (Chapter 17) that a k-bit binary counter can be incremented in $O(1)$ amortized time (bit changes). Suppose we want to decrement the counter as well as increment it. Prove that these two operations cannot be done in $O(1)$ amortized time.

5. **Fibonacci Heaps**

We decide that we do not want to bother with the mark bits in Fibonacci heaps, so we change the cascading-cut rule to cut a node from its parent as soon as it loses its *first* child. We want to prove that we still have $D(n) = O(\log n)$. (This is based on exercise 19.4-2 on page 526 of CLRS3, which Professor Reingold suggested as a good study problem in an email message on October 23.)

(a) Prove an analogue of Lemma 19.1, namely

New Lemma 19.1: Let x be any node in a (modified) Fibonacci heap with $\text{DEGREE}(x) = k$, and children $y_1, y_2, \ldots, y_k$ (in the order in which they were linked to x, from earliest to latest). Then $\text{DEGREE}(y_i) \geq i - 1$, for $1 \leq i \leq k$.

(b) Prove an analogue of Lemma 19.4, namely

New Lemma 19.4: Let x be any node in a (modified) Fibonacci heap. Then $\text{SIZE}(x) \geq 2^{\text{DEGREE}(x)}$.

(c) Prove that the maximum degree $D(n)$ of an node in an n-node (modified) Fibonacci heap is $\lfloor \lg n \rfloor$.