Illinois Institute of Technology
Department of Computer Science

First Examination

CS 430 Introduction to Algorithms
Spring, 2017

Monday, February 20, 2017
10am–11:15am, 002 Herman Hall
11:25am–12:40pm, 104 Rettaliata Engineering Center

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. **Time Bounds**

   Suppose you are given five algorithms with the following exact running times in terms of the input size n:

   (a) $T(n) = T(n-1) + 10$

   (b) $T(n) = T(n-1) + n^2$

   (c) $T(n) = T(n-1) + T(n-2)$

   (d) $T(n) = T(n/2) + 1/n$

   How do the running times of these algorithms change if you double the size of the input? Justify your answers.

2. **Searching Intervals**

You are given a sequence of n intervals on the real line, $I_i = [a_i, b_i]$, $a_i < b_i$, $1 \leq i \leq n$; the intervals are given to you in two separate sorted lists—in the first list, S the intervals are sorted in order by their *starting points*, that is, $a_i < a_{i+1}$ for $1 \leq i < n$, and in the second list E the intervals are sorted by their *ending points*, that is $b_i < b_{i+1}$. Give and analyze a $O(n)$ time algorithm to find the maximum nesting depth of the intervals.

(*Hint*: Consider a merge of S and E in which a counter is kept as the starting points from the list S are merged with the ending points from the list E.)

3. **Restricted-Comparison Sorting**

You are given an array of n elements a_i that is sorted in to decreasing order. You want to sort it into increasing order, but can only swap elements that are separated by one element; that is, you can swap a_i and a_j if and only if $i = j \pm 2$.

(a) Prove that you cannot sort the elements if n is even.

(b) Give an algorithm to sort the items for odd n.

(c) Analyze the number of swaps used in the worst case of your algorithm in the previous part.

(d) Prove that no such restricted-comparison algorithm can sort the elements in fewer than $(n-1)^2/4$ swaps.

(*Hint*: Because we are restricted to adjacent swaps in the even- or odd-indexed elements, we need at least as many swaps as there are inversions in the even- or odd-indexed elements.)

4. **Kraft's Inequality**

   The inequality states that if $l_1, l_2, \ldots, l_n$ are the depths of the n leaves in a binary tree, then

   $$\sum_{i=1}^{n} 2^{-l_i} \leq 1.$$

   Prove this by mathematical induction. (This was suggested as a good exam problem in the lecture on Monday, January 30.)

5. **Binary Search Trees**

   Show that any binary tree with $n \geq 0$ internal nodes and minimal external path length can be colored to make it a proper red-black tree.