CS 373: Combinatorial Algorithms, Fall 2002

Homework 0, due September 5, 2002 at the beginning of class

Name:		
Net ID:	Alias:	U G

Neatly print your name (first name first, with no comma), your network ID, and an alias of your choice into the boxes above. Circle U if you are an undergraduate, and G if you are a graduate student. **Do not sign your name. Do not write your Social Security number.** Staple this sheet of paper to the top of your homework.

Grades will be listed on the course web site by alias give us, so your alias should not resemble your name or your Net ID. If you don't give yourself an alias, we'll give you one that you won't like.

Before you do anything else, please read the Homework Instructions and FAQ on the CS 373 course web page (http://www-courses.cs.uiuc.edu/~cs373/hwx/faq.html) and then check the box below. There are 300 students in CS 373 this semester; we are quite serious about giving zeros to homeworks that don't follow the instructions.

☐ I have read the CS 373 Homework Instructions and FAQ.

Every CS 373 homework has the same basic structure. There are six required problems, some with several subproblems. Each problem is worth 10 points. Only graduate students are required to answer problem 6; undergraduates can turn in a solution for extra credit. There are several practice problems at the end. Stars indicate problems we think are hard.

This homework tests your familiarity with the prerequisite material from CS 173, CS 225, and CS 273, primarily to help you identify gaps in your knowledge. **You are responsible for filling those gaps on your own.** Rosen (the 173/273 textbook), CLRS (especially Chapters 1–7, 10, 12, and A–C), and the lecture notes on recurrences should be sufficient review, but you may want to consult other texts as well.

Required Problems

1. Sort the following functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. Please *don't* turn in proofs, but you should do them anyway to make sure you're right (and for practice).

$$\begin{array}{ccccc} 1 & n & n^2 & \lg n & n \lg n \\ n^{\lg n} & (\lg n)^n & (\lg n)^{\lg n} & n^{\lg \lg n} & n^{1/\lg n} \\ \log_{1000} n & \lg^{1000} n & \lg^{(1000)} n & \lg(n^{1000}) & \left(1 + \frac{1}{1000}\right)^n \end{array}$$

   To simplify notation, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$ and $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the functions n^2, n, $\binom{n}{2}$, n^3 could be sorted either as $n \ll n^2 \equiv \binom{n}{2} \ll n^3$ or as $n \ll \binom{n}{2} \equiv n^2 \ll n^3$.

2. Solve these recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. Please *don't* turn in proofs, but you should do them anyway just for practice. Assume reasonable but nontrivial base cases, and state them if they affect your solution. Extra credit will be given for more exact solutions. *[Hint: Most of these are very easy.]*

$$\begin{array}{ll} A(n) = 2A(n/2) + n & F(n) = 9F(\lfloor n/3 \rfloor + 9) + n^2 \\ B(n) = 3B(n/2) + n & G(n) = 3G(n-1)/5G(n-2) \\ C(n) = 2C(n/3) + n & H(n) = 2H(\sqrt{n}) + 1 \\ D(n) = 2D(n-1) + 1 & I(n) = \min_{1 \le k \le n/2} \big(I(k) + I(n-k) + k\big) \\ E(n) = \max_{1 \le k \le n/2} \big(E(k) + E(n-k) + n\big) & {}^*J(n) = \max_{1 \le k \le n/2} \big(J(k) + J(n-k) + k\big) \end{array}$$

3. Recall that a binary tree is *full* if every node has either two children (an internal node) or no children (a leaf). Give at least *four different* proofs of the following fact:

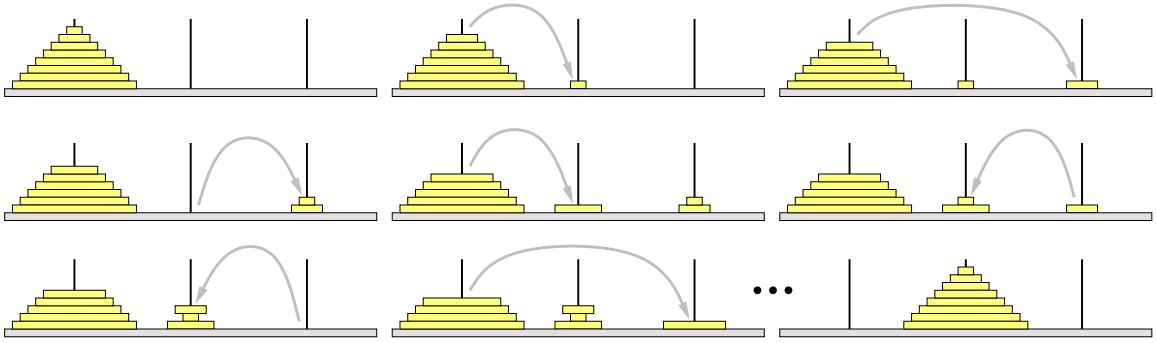
   > *In any full binary tree, the number of leaves is exactly one more than the number of internal nodes.*

   For full credit, each proof must be self-contained, the proof must be substantially different from each other, and at least one proof must *not* use induction. For each n, your nth correct proof is worth n points, so you need four proofs to get full credit. Each correct proof beyond the fourth earns you extra credit. *[Hint: I know of at least six different proofs.]*

4. Most of you are probably familiar with the story behind the Tower of Hanoï puzzle:[1]

> At the great temple of Benares, there is a brass plate on which three vertical diamond shafts are fixed. On the shafts are mounted n golden disks of decreasing size.[2] At the time of creation, the god Brahma placed all of the disks on one pin, in order of size with the largest at the bottom. The Hindu priests unceasingly transfer the disks from peg to peg, one at a time, never placing a larger disk on a smaller one. When all of the disks have been transferred to the last pin, the universe will end.

Recently the temple at Benares was relocated to southern California, where the monks are considerably more laid back about their job. At the "Towers of Hollywood", the golden disks were replaced with painted plywood, and the diamond shafts were replaced with Plexiglas. More importantly, the restriction on the order of the disks was relaxed. While the disks are being moved, the *bottom* disk on any pin must be the *largest* disk on that pin, but disks further up in the stack can be in any order. However, after all the disks have been moved, they must be in sorted order again.



The Towers of Hollywood.

Describe an algorithm[3] that moves a stack of n disks from one pin to the another using the smallest possible number of moves. For full credit, your algorithm should be non-recursive, but a recursive algorithm is worth significant partial credit. *Exactly* how many moves does your algorithm perform? *[Hint: The Hollywood monks can bring about the end of the universe quite a bit faster than the original monks at Benares could.]*

The problem of computing the minimum number of moves was posed in the most recent issue of the *American Mathematical Monthly* (August/September 2002). No solution has been published yet.

[1] The puzzle and the accompanying story were both invented by the French mathematician Eduoard Lucas in 1883. See http://www.cs.wm.edu/~pkstoc/toh.html

[2] In the original legend, $n = 64$. In the 1883 wooden puzzle, $n = 8$.

[3] Since you've read the Homework Instructions, you know exactly what this phrase means.

5. On their long journey from Denmark to England, Rosencrantz and Guildenstern amuse themselves by playing the following game with a fair coin. First Rosencrantz flips the coin over and over until it comes up tails. Then Guildenstern flips the coin over and over until he gets as many heads in a row as Rosencrantz got on his turn. Here are three typical games:

   Rosencrantz: H H T
   Guildenstern: H T H H

   Rosencrantz: T
   Guildenstern: (no flips)

   Rosencrantz: H H H T
   Guildenstern: T H H T H H T H T H H H

   (a) What is the expected number of flips in one of Rosencrantz's turns?

   (b) Suppose Rosencrantz flips k heads in a row on his turn. What is the expected number of flips in Guildenstern's next turn?

   (c) What is the expected total number of flips (by both Rosencrantz and Guildenstern) in a single game?

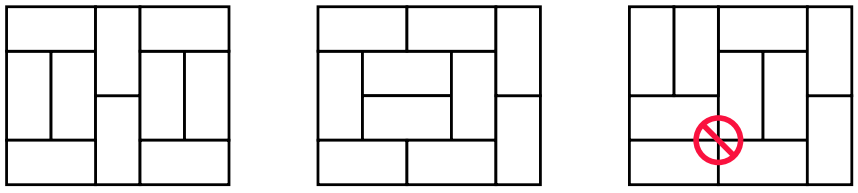
   Prove your answers are correct. If you have to appeal to "intuition" or "common sense", your answer is almost certainly wrong! You must give exact answers for full credit, but asymptotic bounds are worth significant partial credit.

6. *[This problem is required only for graduate students (including I2CS students); undergrads can submit a solution for extra credit.]*

   Tatami are rectangular mats used to tile floors in traditional Japanese houses. Exact dimensions of tatami mats vary from one region of Japan to the next, but they are always twice as long in one dimension than in the other. (In Tokyo, the standard size is 180cm×90cm.)

   (a) How many different ways are there to tile a $2 \times n$ rectangular room with 1×2 tatami mats? Set up a recurrence and derive an *exact* closed-form solution. *[Hint: The answer involves a familiar recursive sequence.]*

   (b) According to tradition, tatami mats are always arranged so that four corners never meet. Thus, the first two arrangements below are traditional, but not the third.



Two traditional tatami arrangements and one non-traditional arrangement.

   How many different *traditional* ways are there to tile a $3 \times n$ rectangular room with 1×2 tatami mats? Set up a recurrence and derive an *exact* closed-form solution.

   *(c) **[5 points extra credit]** How many different *traditional* ways are there to tile an $n \times n$ square with 1×2 tatami mats? Prove your answer is correct.

Practice Problems

These problems are only for your benefit; other problems can be found in previous semesters' homeworks on the course web site. You are *strongly* encouraged to do some of these problems as additional practice. Think of them as potential exam questions (hint, hint). Feel free to ask about any of these questions on the course newsgroup, during office hours, or during review sessions.

1. Removing any edge from a binary tree with n nodes partitions it into two smaller binary trees. If both trees have at least $\lceil (n-1)/3 \rceil$ nodes, we say that the partition is *balanced*.

   (a) Prove that every binary tree with more than one vertex has a balanced partition. *[Hint: I know of at least two different proofs.]*

   (b) If each smaller tree has more than $\lfloor n/3 \rfloor$ nodes, we say that the partition is *strictly balanced*. Show that for every n, there is an n-node binary tree with *no* strictly balanced partition.

2. Describe an algorithm CountToTenToThe(n) that prints the integers from 1 to 10^n.

   Assume you have a subroutine PrintDigit(d) that prints any integer d between 0 and 9, and another subroutine PrintSpace that prints a space character. Both subroutines run in $O(1)$ time. You may want to write (and analyze) a separate subroutine PrintInteger to print an arbitrary integer.

   Since integer variables cannot store arbitrarily large values in most programming languages, your algorithm must not store any value larger than $\max\{10, n\}$ in any single integer variable. Thus, the following algorithm is **not correct**:

   ```
   BogusCountToTenToThe(n):
     for i ← 1 to Power(10, n)
       PrintInteger(i)
   ```

   (So what exactly *can* you pass to PrintInteger?)

   What is the running time of your algorithm (as a function of n)? How many digits and spaces does it print? How much space does it use?

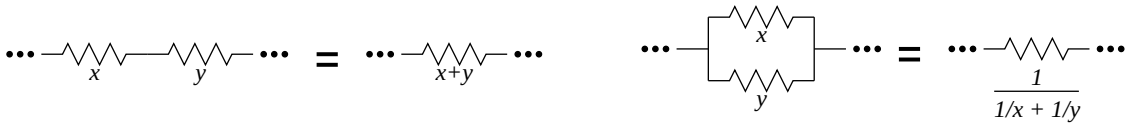
3. I'm sure you remember the following simple rules for taking derivatives:

   - Simple cases: $\frac{d}{dx}\alpha = 0$ for any constant α, and $\frac{d}{dx}x = 1$
   - Linearity: $\frac{d}{dx}(f(x) + g(x)) = f'(x) + g'(x)$
   - The product rule: $\frac{d}{dx}(f(x) \cdot g(x)) = f'(x) \cdot g(x) + f(x) \cdot g'(x)$
   - The chain rule: $\frac{d}{dx}(f(g(x)) = f'(g(x)) \cdot g'(x)$

   Using *only* these rules and induction, prove that $\frac{d}{dx}x^c = cx^{c-1}$ for any integer $c \neq -1$. Do not use limits, integrals, or any other concepts from calculus, except for the simple identities listed above. *[Hint: Don't forget about negative values of c!]*

4. This problem asks you to calculate the total resistance between two points in a series-parallel resistor network. Don't worry if you haven't taken a circuits class; everything you need to know can be summed up in two sentences and a picture.

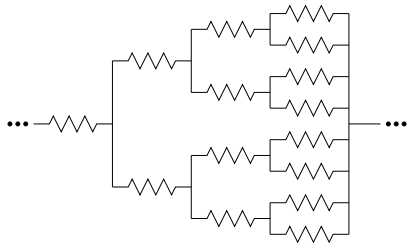
   - The total resistance of two resistors in *series* is the sum of their individual resistances.
   - The total resistance of two resistors in *parallel* is the reciprocal of the sum of the reciprocals of their individual resistances.



Equivalence laws for series-parallel resistor networks.

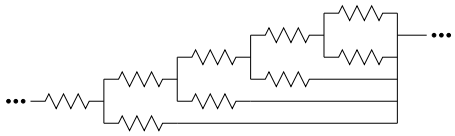
What is the *exact* total resistance[4] of the following resistor networks as a function of n? Prove your answers are correct. *[Hint: Use induction. Duh.]*

(a) A complete binary tree with depth n, with a 1Ω resistor at every node, and a common wire joining all the leaves. Resistance is measured between the root and the leaves.



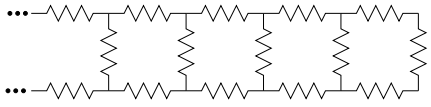
A balanced binary resistor tree with depth 3.

(b) A totally unbalanced full binary tree with depth n (every internal node has two children, one of which is a leaf) with a 1Ω resistor at every node, and a common wire joining all the leaves. Resistance is measured between the root and the leaves.



A totally unbalanced binary resistor tree with depth 4.

*(c) A ladder with n rungs, with a 1Ω resistor on every edge. Resistance is measured between the bottom of the legs.



A resistor ladder with 5 rungs.

[4]The ISO standard unit of resistance is the Ohm, written with the symbol Ω. Don't confuse this with the asymptotic notation $\Omega(f(n))$!