CS 473U: Undergraduate Algorithms, Fall 2006
Homework 0

Due Friday, September 1, 2006 at noon in 3229 Siebel Center

Name:	
Net ID:	Alias:

☐ I understand the Homework Instructions and FAQ.

- Neatly print your full name, your NetID, and an alias of your choice in the boxes above, and submit this page with your solutions. We will list homework and exam grades on the course web site by alias. For privacy reasons, your alias should not resemble your name, your NetID, your university ID number, or (God forbid) your Social Security number. Please use the same alias for every homework and exam.

  Federal law forbids us from publishing your grades, even anonymously, without your explicit permission. **By providing an alias, you grant us permission to list your grades on the course web site; if you do not provide an alias, your grades will not be listed.**

- Please carefully read the Homework Instructions and FAQ on the course web page, and then check the box above. This page describes what we expect in your homework solutions—start each numbered problem on a new sheet of paper, write your name and NetID on every page, don't turn in source code, analyze and prove everything, use good English and good logic, and so on—as well as policies on grading standards, regrading, and plagiarism. **See especially the policies regarding the magic phrases "I don't know" and "and so on".** If you have *any* questions, post them to the course newsgroup or ask in lecture.

- This homework tests your familiarity with prerequisite material—basic data structures, big-Oh notation, recurrences, discrete probability, and most importantly, induction—to help you identify gaps in your knowledge. **You are responsible for filling those gaps on your own.** Each numbered problem is worth 10 points; not all subproblems have equal weight.

#	1	2	3	4	5	6*	Total
Score							
Grader							

Please put your answers to problems 1 and 2 on the same page.

1. Sort the functions listed below from asymptotically smallest to asymptotically largest, indicating ties if there are any. **Do not turn in proofs**, but you should probably do them anyway, just for practice.

   To simplify your answers, write $f(n) \ll g(n)$ to mean $f(n) = o(g(n))$, and write $f(n) \equiv g(n)$ 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 $n \ll \binom{n}{2} \equiv n^2 \ll n^3$.

   $$\begin{array}{cccccccc} \lg n & \ln n & \sqrt{n} & n & n \lg n & n^2 & 2^n & n^{1/n} \\ n^{1+1/\lg n} & \lg^{1000} n & 2^{\sqrt{\lg n}} & (\sqrt{2})^{\lg n} & \lg^{\sqrt{2}} n & n^{\sqrt{2}} & (1+\tfrac{1}{n})^n & n^{1/1000} \\ H_n & H_{\sqrt{n}} & 2^{H_n} & H_{2^n} & F_n & F_{n/2} & \lg F_n & F_{\lg n} \end{array}$$

   In case you've forgotten:

   - $\lg n = \log_2 n \neq \ln n = \log_e n$
   - $\lg^3 n = (\lg n)^3 \neq \lg \lg \lg n$.
   - The harmonic numbers: $H_n = \sum_{i=1}^{n} 1/i \approx \ln n + 0.577215\ldots$
   - The Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$

2. Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. Proofs are *not* required; just give us the list of answers. **Don't turn in proofs**, but you should do them anyway, just for practice. Assume reasonable but nontrivial base cases. **If your solution requires specific base cases, state them.** Extra credit will be awarded for more exact solutions.

   (a) $A(n) = 2A(n/4) + \sqrt{n}$

   (b) $B(n) = 3B(n/3) + n/\lg n$

   (c) $C(n) = \dfrac{2C(n-1)}{C(n-2)}$ *[Hint: This is easy!]*

   (d) $D(n) = D(n-1) + 1/n$

   (e) $E(n) = E(n/2) + D(n)$

   (f) $F(n) = 4F\left(\left\lceil \dfrac{n-8}{2} \right\rceil + \left\lfloor \dfrac{3n}{\log_\pi n} \right\rfloor\right) + 6\binom{n+5}{2} - 42n\lg^7 n + \sqrt{13n-6} + \dfrac{\lg\lg n + 1}{\lg n \lg\lg n \lg\lg\lg n}$

   (g) $G(n) = 2G(n-1) - G(n-2) + n$

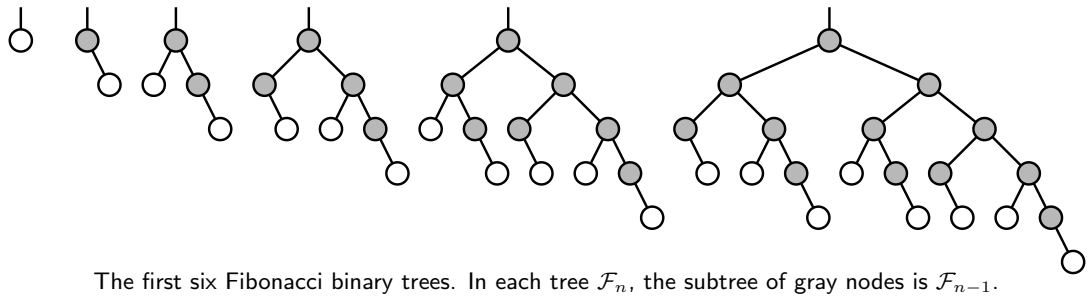
   (h) $H(n) = 2H(n/2) - 2H(n/4) + 2^n$

   (i) $I(n) = I(n/2) + I(n/4) + I(n/6) + I(n/12) + n$

   ★(j) $J(n) = \sqrt{n} \cdot J(2\sqrt{n}) + n$
   [Hint: First solve the secondary recurrence $j(n) = 1 + j(2\sqrt{n})$.]

3. The nth *Fibonacci binary tree* $\mathcal{F}_n$ is defined recursively as follows:

   - $\mathcal{F}_1$ is a single root node with no children.
   - For all $n \geq 2$, $\mathcal{F}_n$ is obtained from $\mathcal{F}_{n-1}$ by adding a right child to every leaf and adding a left child to every node that has only one child.



The first six Fibonacci binary trees. In each tree $\mathcal{F}_n$, the subtree of gray nodes is $\mathcal{F}_{n-1}$.

   (a) Prove that the number of leaves in $\mathcal{F}_n$ is precisely the nth Fibonacci number: $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$.

   (b) How many nodes does $\mathcal{F}_n$ have? For full credit, give an *exact*, closed-form answer in terms of Fibonacci numbers, and prove your answer is correct.

   (c) Prove that the left subtree of $\mathcal{F}_n$ is a copy of $\mathcal{F}_{n-2}$.

4. Describe and analyze a data structure that stores set of n records, each with a numerical *key* and a numerical *priority*, such that the following operation can be performed quickly:

   RANGETOP(a, z) : return the highest-priority record whose key is between a and z.

   For example, if the (key, priority) pairs are

   $$(3,1),\ (4,9),\ (9,2),\ (6,3),\ (5,8),\ (7,5),\ (1,4),\ (0,7),$$

   then RANGETOP$(2, 8)$ would return the record with key 4 and priority 9 (the second record in the list).

   You may assume that no two records have equal keys or equal priorities, and that no record has a key equal to a or z. Analyze both the size of your data structure and the running time of your RANGETOP algorithm. For full credit, your data structure must be as small as possible *and* your RANGETOP algorithm must be as fast as possible.

   [Hint: How would you compute the number *of keys between a and z? How would you solve the problem if you knew that a is always $-\infty$?]*

5. Penn and Teller agree to play the following game. Penn shuffles a standard deck[1] of playing cards so that every permutation is equally likely. Then Teller draws cards from the deck, one at a time without replacement, until he draws the three of clubs (3♣), at which point the remaining undrawn cards instantly burst into flames.

   The first time Teller draws a card from the deck, he gives it to Penn. From then on, until the game ends, whenever Teller draws a card whose value is smaller than the last card he gave to Penn, he gives the new card to Penn.[2] To make the rules unambiguous, they agree beforehand that $A = 1$, $J = 11$, $Q = 12$, and $K = 13$.

   (a) What is the expected number of cards that Teller draws?

   (b) What is the expected *maximum* value among the cards Teller gives to Penn?

   (c) What is the expected *minimum* value among the cards Teller gives to Penn?

   (d) What is the expected number of cards that Teller gives to Penn?

   Full credit will be given only for *exact* answers (with correct proofs, of course).

*6. **[Extra credit]**[3]

   Lazy binary is a variant of standard binary notation for representing natural numbers where we allow each "bit" to take on one of three values: 0, 1, or 2. Lazy binary notation is defined inductively as follows.

   - The lazy binary representation of zero is 0.
   - Given the lazy binary representation of any non-negative integer n, we can construct the lazy binary representation of $n + 1$ as follows:
     (a) increment the rightmost digit;
     (b) if any digit is equal to 2, replace the rightmost 2 with 0 and increment the digit immediately to its left.

   Here are the first several natural numbers in lazy binary notation:

   0, 1, 10, 11, 20, 101, 110, 111, 120, 201, 210, 1011, 1020, 1101, 1110, 1111, 1120, 1201, 1210, 2011, 2020, 2101, 2110, 10111, 10120, 10201, 10210, 11011, 11020, 11101, 11110, 11111, 11120, 11201, 11210, 12011, 12020, 12101, 12110, 20111, 20120, 20201, 20210, 21011, 21020, 21101, 21110, 101111, 101120, 101201, 101210, 102011, 102020, 102101, 102110, ...

   (a) Prove that in any lazy binary number, between any two 2s there is at least one 0, and between two 0s there is at least one 2.

   (b) Prove that for any natural number N, the sum of the digits of the lazy binary representation of N is exactly $\lfloor \lg(N + 1) \rfloor$.

[1] In a standard deck of 52 cards, each card has a *suit* in the set $\{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$ and a *value* in the set $\{A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}$, and every possible suit-value pair appears in the deck exactly once. Actually, to make the game more interesting, Penn and Teller normally use razor-sharp ninja throwing cards.

[2] Specifically, he hurls them from the opposite side of the stage directly into the back of Penn's right hand.

[3] The "I don't know" rule does not apply to extra credit problems. There is no such thing as "partial extra credit".