CS 373: Combinatorial Algorithms, Spring 2001

Homework 0, due January 23, 2001 at the beginning of class

Name:	
Net ID:	Alias:

Neatly print your name (first name first, with no comma), your network ID, and a short alias into the boxes above. **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.

This homework tests your familiarity with the prerequisite material from CS 173, CS 225, and CS 273—many of these problems have appeared on homeworks or exams in those classes—primarily to help you identify gaps in your knowledge. **You are responsible for filling those gaps on your own.** Parberry and Chapters 1–6 of CLR should be sufficient review, but you may want to consult other texts as well.

Before you do anything else, read the Homework Instructions and FAQ on the CS 373 course web page (http://www-courses.cs.uiuc.edu/~cs373/hw/faq.html), and then check the box below. This web page gives instructions on how to write and submit homeworks—staple your solutions together in order, write your name and netID on every page, don't turn in source code, analyze everything, use good English and good logic, and so forth.

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

Required Problems

1. (a) Prove that any positive integer can be written as the sum of distinct powers of 2. For example: $42 = 2^5 + 2^3 + 2^1$, $25 = 2^4 + 2^3 + 2^0$, $17 = 2^4 + 2^0$. *[Hint: 'Write the number in binary' is* not *a proof; it just restates the problem.]*

   (b) Prove that any positive integer can be written as the sum of distinct *nonconsecutive* Fibonacci numbers—if F_n appears in the sum, then neither F_{n+1} nor F_{n-1} will. For example: $42 = F_9 + F_6$, $25 = F_8 + F_4 + F_2$, $17 = F_7 + F_4 + F_2$.

   (c) Prove that *any* integer (positive, negative, or zero) can be written in the form $\sum_i \pm 3^i$, where the exponents i are distinct non-negative integers. For example: $42 = 3^4 - 3^3 - 3^2 - 3^1$, $25 = 3^3 - 3^1 + 3^0$, $17 = 3^3 - 3^2 - 3^0$.

2. Sort the following 20 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice.

1	n	n^2	$\lg n$	$\lg^* n$
$2^{2^{\lg\lg n+1}}$	$\lg^* 2^n$	$2^{\lg^* n}$	$\lfloor \lg(n!) \rfloor$	$\lfloor \lg n \rfloor !$
$n^{\lg n}$	$(\lg n)^n$	$(\lg n)^{\lg n}$	$n^{1/\lg n}$	$n^{\lg\lg n}$
$\log_{1000} n$	$\lg^{1000} n$	$\lg^{(1000)} n$	$\left(1+\frac{1}{1000}\right)^n$	$n^{1/1000}$

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 $n \ll \binom{n}{2} \equiv n^2 \ll n^3$.

3. Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. Extra credit will be given for more exact solutions.

   (a) $A(n) = 5A(n/3) + n \log n$

   (b) $B(n) = \min_{0<k<n} \big(B(k) + B(n-k) + 1\big)$.

   (c) $C(n) = 4C(\lfloor n/2 \rfloor + 5) + n^2$

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

   *(e) $E(n) = n + 2\sqrt{n} \cdot E(\sqrt{n})$

4. This problem asks you to simplify some recursively defined boolean formulas as much as possible. In each case, prove that your answer is correct. Each proof can be just a few sentences long, but it must be a *proof*.

   (a) Suppose $\alpha_0 = p$, $\alpha_1 = q$, and $\alpha_n = (\alpha_{n-2} \wedge \alpha_{n-1})$ for all $n \geq 2$. Simplify α_n as much as possible. *[Hint: What is α_5?]*

   (b) Suppose $\beta_0 = p$, $\beta_1 = q$, and $\beta_n = (\beta_{n-2} \Leftrightarrow \beta_{n-1})$ for all $n \geq 2$. Simplify β_n as much as possible. *[Hint: What is β_5?]*

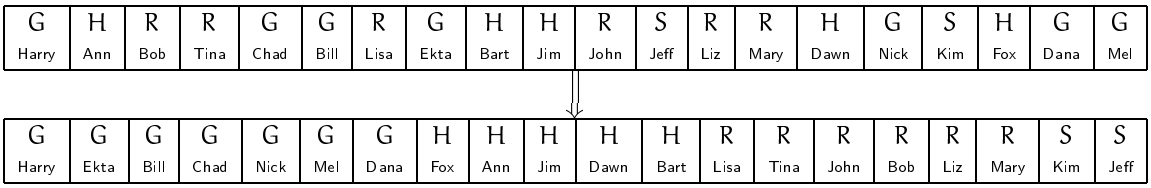
   (c) Suppose $\gamma_0 = p$, $\gamma_1 = q$, and $\gamma_n = (\gamma_{n-2} \Rightarrow \gamma_{n-1})$ for all $n \geq 2$. Simplify γ_n as much as possible. *[Hint: What is γ_5?]*

   (d) Suppose $\delta_0 = p$, $\delta_1 = q$, and $\delta_n = (\delta_{n-2} \bowtie \delta_{n-1})$ for all $n \geq 2$, where $\bowtie$ is some boolean function with two arguments. Find a boolean function $\bowtie$ such that $\delta_n = \delta_m$ if and only if $n - m$ is a multiple of 4. *[Hint: There is only one such function.]*

5. Every year, upon their arrival at Hogwarts School of Witchcraft and Wizardry, new students are sorted into one of four houses (Gryffindor, Hufflepuff, Ravenclaw, or Slytherin) by the Hogwarts Sorting Hat. The student puts the Hat on their head, and the Hat tells the student which house they will join. This year, a failed experiment by Fred and George Weasley filled almost all of Hogwarts with sticky brown goo, mere moments before the annual Sorting. As a result, the Sorting had to take place in the basement hallways, where there was so little room to move that the students had to stand in a long line.

   After everyone learned what house they were in, the students tried to group together by house, but there was too little room in the hallway for more than one student to move at a time. Fortunately, the Sorting Hat took CS 373 many years ago, so it knew how to group the students as quickly as possible. What method did the Sorting Hat use?

   More formally, you are given an array of n items, where each item has one of four possible values, possibly with a pointer to some additional data. Describe an algorithm[1] that rearranges the items into four clusters in $O(n)$ time using only $O(1)$ extra space.



6. *[This problem is required only for graduate students taking CS 373 for a full unit; anyone else can submit a solution for extra credit.]*

   Penn and Teller have a special deck of fifty-two cards, with no face cards and nothing but clubs—the ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ..., 52 of clubs. (They're big cards.) Penn shuffles the deck until each each of the 52! possible orderings of the cards is equally likely. He then takes cards one at a time from the top of the deck and gives them to Teller, stopping as soon as he gives Teller the three of clubs.

   (a) On average, how many cards does Penn give Teller?

   (b) On average, what is the smallest-numbered card that Penn gives Teller?

   *(c) On average, what is the largest-numbered card that Penn gives Teller?

   [Hint: Solve for an n-card deck and then set $n = 52$.] In each case, give *exact* answers and prove that they are correct. If you have to appeal to "intuition" or "common sense", your answers are probably wrong!

[1] Since you've read the Homework Instructions, you know what the phrase 'describe an algorithm' means. Right?

Practice Problems

The remaining problems are entirely for your benefit; similar questions will appear in every homework. Don't turn in solutions—we'll just throw them out—but feel free to ask us about practice questions during office hours and review sessions. Think of them as potential exam questions (hint, hint). We'll post solutions to *some* of the practice problems after the homeworks are due.

1. Recall the standard recursive definition of the Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. Prove the following identities for all positive integers n and m.

   (a) F_n is even if and only if n is divisible by 3.

   (b) $\sum_{i=0}^{n} F_i = F_{n+2} - 1$

   (c) $F_n^2 - F_{n+1}F_{n-1} = (-1)^{n+1}$

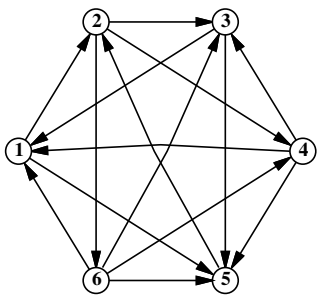
   ★(d) If n is an integer multiple of m, then F_n is an integer multiple of F_m.

2. (a) Prove the following identity by induction:

$$\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}\binom{n}{n-k}.$$

   (b) Give a non-inductive combinatorial proof of the same identity, by showing that the two sides of the equation count exactly the same thing in two different ways. There is a correct one-sentence proof.

3. A *tournament* is a directed graph with exactly one edge between every pair of vertices. (Think of the nodes as players in a round-robin tournament, where each edge points from the winner to the loser.) A *Hamiltonian path* is a sequence of directed edges, joined end to end, that visits every vertex exactly once. Prove that every tournament contains at least one Hamiltonian path.



A six-vertex tournament containing the Hamiltonian path $6 \to 4 \to 5 \to 2 \to 3 \to 1$.

4. Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. You do not need to turn in proofs (in fact, please *don't* turn in proofs), but you should do them anyway just for practice. Assume reasonable but nontrivial base cases if none are supplied. Extra credit will be given for more exact solutions.

   (a) $A(n) = A(n/2) + n$

   (b) $B(n) = 2B(n/2) + n$

   ★(c) $C(n) = n + \frac{1}{2}\big(C(n-1) + C(3n/4)\big)$

   (d) $D(n) = \max_{n/3<k<2n/3} \big(D(k) + D(n-k) + n\big)$

   *(e) $E(n) = 2E(n/2) + n/\lg n$

   *(f) $F(n) = \dfrac{F(n-1)}{F(n-2)}$, where $F(1) = 1$ and $F(2) = 2$.

   *(g) $G(n) = G(n/2) + G(n/4) + G(n/6) + G(n/12) + n$ [Hint: $\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{12} = 1$.]

   *(h) $H(n) = n + \sqrt{n} \cdot H(\sqrt{n})$

   *(i) $I(n) = (n-1)(I(n-1) + I(n-2))$, where $F(0) = F(1) = 1$

   *(j) $J(n) = 8J(n-1) - 15J(n-2) + 1$

5. (a) Prove that $2^{\lceil \lg n \rceil + \lfloor \lg n \rfloor} = \Theta(n^2)$.

   (b) Prove or disprove: $2^{\lfloor \lg n \rfloor} = \Theta\big(2^{\lceil \lg n \rceil}\big)$.

   (c) Prove or disprove: $2^{2^{\lfloor \lg \lg n \rfloor}} = \Theta\big(2^{2^{\lceil \lg \lg n \rceil}}\big)$.

   (d) Prove or disprove: If $f(n) = O(g(n))$, then $\log(f(n)) = O(\log(g(n)))$.

   (e) Prove or disprove: If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$.

   *(f) Prove that $\log^k n = o(n^{1/k})$ for any positive integer k.

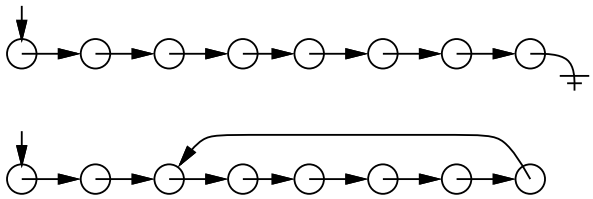
6. Evaluate the following summations; simplify your answers as much as possible. Significant partial credit will be given for answers in the form $\Theta(f(n))$ for some recognizable function $f(n)$.

   (a) $\displaystyle\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i} \frac{1}{i}$

   *(b) $\displaystyle\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i} \frac{1}{j}$

   (c) $\displaystyle\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{i} \frac{1}{k}$

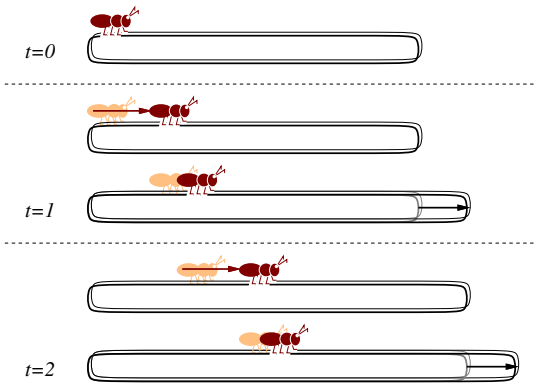
7. Suppose you have a pointer to the head of singly linked list. Normally, each node in the list only has a pointer to the next element, and the last node's pointer is NULL. Unfortunately, your list might have been corrupted by a bug in somebody else's code[2], so that the last node has a pointer back to some other node in the list instead.



Top: A standard linked list. Bottom: A corrupted linked list.

Describe an algorithm that determines whether the linked list is corrupted or not. Your algorithm must not modify the list. For full credit, your algorithm should run in $O(n)$ time, where n is the number of nodes in the list, and use $O(1)$ extra space (not counting the list itself).

*8. An ant is walking along a rubber band, starting at the left end. Once every second, the ant walks one inch to the right, and then you make the rubber band one inch longer by pulling on the right end. The rubber band stretches uniformly, so stretching the rubber band also pulls the ant to the right. The initial length of the rubber band is n inches, so after t seconds, the rubber band is $n + t$ inches long.



Every second, the ant walks an inch, and then the rubber band is stretched an inch longer.

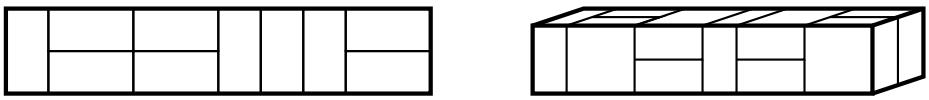
(a) How far has the ant moved after t seconds, as a function of n and t? Set up a recurrence and (for full credit) give an *exact* closed-form solution. [Hint: What *fraction* of the rubber band's length has the ant walked?]

(b) How long does it take the ant to get to the right end of the rubber band? For full credit, give an answer of the form $f(n) + \Theta(1)$ for some explicit function $f(n)$.

[2] After all, *your* code is always completely 100% bug-free. Isn't that right, Mr. Gates?

9. (a) A *domino* is a 2×1 or 1×2 rectangle. How many different ways are there to completely fill a $2 \times n$ rectangle with n dominos? Set up a recurrence relation and give an *exact* closed-form solution.

   (b) A *slab* is a three-dimensional box with dimensions $1 \times 2 \times 2$, $2 \times 1 \times 2$, or $2 \times 2 \times 1$. How many different ways are there to fill a $2 \times 2 \times n$ box with n slabs? Set up a recurrence relation and give an *exact* closed-form solution.



A 2×10 rectangle filled with ten dominos, and a $2 \times 2 \times 10$ box filled with ten slabs.

10. Professor George O'Jungle has a favorite 26-node binary tree, whose nodes are labeled by letters of the alphabet. The preorder and postorder sequences of nodes are as follows:

    preorder: M N H C R S K W T G D X I Y A J P O E Z V B U L Q F
    postorder: C W T K S G R H D N A O E P J Y Z I B Q L F U V X M

    Draw Professor O'Jungle's binary tree, and give the inorder sequence of nodes.

11. Alice and Bob each have a fair n-sided die. Alice rolls her die once. Bob then repeatedly throws his die until he rolls a number at least as big as the number Alice rolled. Each time Bob rolls, he pays Alice \$1. (For example, if Alice rolls a 5, and Bob rolls a 4, then a 3, then a 1, then a 5, the game ends and Alice gets \$4. If Alice rolls a 1, then no matter what Bob rolls, the game will end immediately, and Alice will get \$1.)

    Exactly how much money does Alice expect to win at this game? Prove that your answer is correct. If you have to appeal to 'intuition' or 'common sense', your answer is probably wrong!

12. Prove that for any nonnegative parameters a and b, the following algorithms terminate and produce identical output.

```
SLOWEUCLID(a, b):
  if b > a
      return SLOWEUCLID(b, a)
  else if b = 0
      return a
  else
      return SLOWEUCLID(b, a − b)
```

```
FASTEUCLID(a, b):
  if b = 0
      return a
  else
      return FASTEUCLID(b, a mod b)
```