CS 373: Combinatorial Algorithms, Fall 2000
Midterm 1 — October 3, 2000

Name:		
Net ID:	Alias:	U $^3\!/_4$ 1

This is a closed-book, closed-notes exam!

If you brought anything with you besides writing instruments and your $8\frac{1}{2}'' \times 11''$ cheat sheet, please leave it at the front of the classroom.

- Print your name, netid, and alias in the boxes above. Circle U if you are an undergrad, $^3\!/_4$ if you are a 3/4-unit grad student, or 1 if you are a 1-unit grad student. Print your name at the top of every page (in case the staple falls out!).
- **Answer four of the five questions on the exam.** Each question is worth 10 points. If you answer more than four questions, the one with the lowest score will be ignored. **1-unit graduate students must answer question 5.**
- Please write your final answers on the front of the exam pages. Use the backs of the pages as scratch paper. Let us know if you need more paper.
- Unless we specifically say otherwise, proofs are not required. However, they may help us give you partial credit.
- Read the entire exam before writing anything. Make sure you understand what the questions are asking. If you give a beautiful answer to the wrong question, you'll get no credit. If any question is unclear, please ask one of us for clarification.
- Don't spend too much time on any single problem. If you get stuck, move on to something else and come back later.
- Write something down for every problem. Don't panic and erase large chunks of work. Even if you think it's absolute nonsense, it might be worth partial credit.
- Relax. Breathe. Kick some ass.

#	Score	Grader
1		
2		
3		
4		
5		
Total		

1. **Multiple Choice**

   Every question below has one of the following answers.

   (a) $\Theta(1)$ (b) $\Theta(\log n)$ (c) $\Theta(n)$ (d) $\Theta(n \log n)$ (e) $\Theta(n^2)$

   For each question, write the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 1 point, but each incorrect answer *costs* you $\frac{1}{2}$ point. You cannot score below zero.

☐ What is $\sum_{i=1}^{n} \log i$?

☐ What is $\sum_{i=1}^{n} \frac{n}{i}$?

☐ How many digits do you need to write 2^n in decimal?

☐ What is the solution of the recurrence $T(n) = 25T(n/5) + n$?

☐ What is the solution of the recurrence $T(n) = T(n-1) + \frac{1}{2^n}$?

☐ What is the solution of the recurrence $T(n) = 3T\left(\left\lceil \frac{n+51}{3} \right\rceil\right) + 17n - \sqrt[7]{\lg \lg n} - 2^{2^{\log^* n}} + \pi$?

☐ What is the worst-case running time of randomized quicksort?

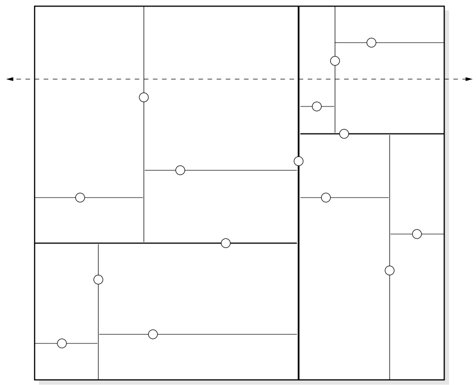
☐ The expected time for inserting one item into an n-node randomized treap is $O(\log n)$. What is the worst-case time for a sequence of n insertions into an initially empty treap?

☐ The amortized time for inserting one item into an n-node scapegoat tree is $O(\log n)$. What is the worst-case time for a sequence of n insertions into an initially empty scapegoat tree?

☐ In the worst case, how many nodes can be in the root list of a Fibonacci heap storing n keys, immediately after a DecreaseKey operation?

☐ Every morning, an Amtrak train leaves Chicago for Champaign, 200 miles away. The train can accelerate or decelerate at 10 miles per hour per second, and it has a maximum speed of 60 miles an hour. Every 50 miles, the train must stop for five minutes while a school bus crosses the tracks. Every hour, the conductor stops the train for a union-mandated 10-minute coffee break. How long does it take the train to reach Champaign?

2. Suppose we have n points scattered inside a two-dimensional box. A *kd-tree* recursively subdivides the rectangle as follows. First we split the box into two smaller boxes with a *vertical* line, then we split each of those boxes with *horizontal* lines, and so on, always alternating between horizontal and vertical splits. Each time we split a box, the splitting line passes through some point inside the box (*not* on the boundary) and partitions the rest of the interior points as evenly as possible. If a box doesn't contain any points, we don't split it any more; these final empty boxes are called *cells*.



A kd-tree for 15 points. The dashed line crosses four cells.

(a) *[2 points]* How many cells are there, as a function of n? Prove your answer is correct.

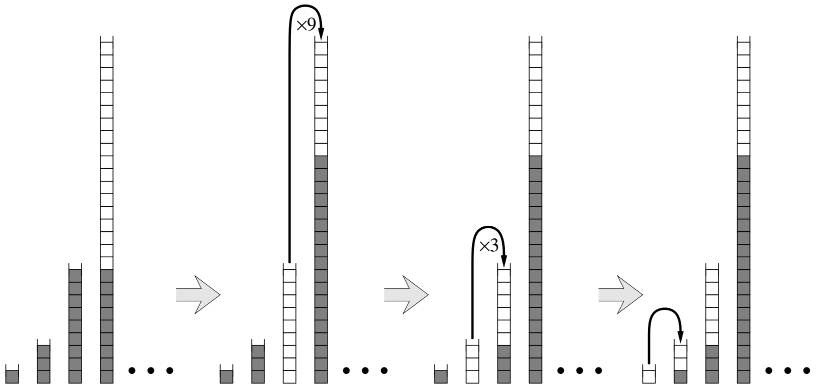
(b) *[8 points]* In the worst case, exactly how many cells can a horizontal line cross, as a function of n? Prove your answer is correct. Assume that $n = 2^k - 1$ for some integer k.
[For full credit, you must give an exact answer. A tight asymptotic bound (with proof) is worth 5 points. A correct recurrence is worth 3 points.]

(c) *[5 points extra credit]* In the worst case, how many cells can a *diagonal* line cross?

2. (continued)

Incidentally, 'kd-tree' originally meant 'k-dimensional tree'—for example, the specific data structure described here used to be called a '2d-tree'—but current usage ignores this etymology. The phrase 'd-dimensional kd-tree' is now considered perfectly standard, even though it's just as redundant as 'ATM machine', 'PIN number', 'HIV virus', 'PDF format', 'Mt. Fujiyama', 'Sahara Desert', 'The La Brea Tar Pits', or 'and etc.' On the other hand, 'BASIC code' is *not* redundant; 'Beginner's All-Purpose Instruction Code' is a backronym. Hey, aren't you supposed to be taking a test?

3. A *multistack* consists of an infinite series of stacks $S_0, S_1, S_2, \ldots$, where the ith stack S_i can hold up to 3^i elements. Whenever a user attempts to push an element onto any full stack S_i, we first move all the elements in S_i to stack S_{i+1} to make room. But if S_{i+1} is already full, we first move all its members to S_{i+2}, and so on. Moving a single element from one stack to the next takes $O(1)$ time.



Making room for one new element in a multistack.

(a) *[1 point]* In the worst case, how long does it take to push one more element onto a multistack containing n elements?

(b) *[9 points]* Prove that the amortized cost of a push operation is $O(\log n)$, where n is the maximum number of elements in the multistack. You can use any method you like.

3. (continued)

4. After graduating with a computer science degree, you find yourself working for a software company that publishes a word processor. The program stores a document containing n characters, grouped into p paragraphs. Your manager asks you to implement a 'Sort Paragraphs' command that rearranges the paragraphs into alphabetical order.

   Design and analyze and efficient paragraph-sorting algorithm, using the following pair of routines as black boxes.

   - CompareParagraphs(i, j) compares the ith and jth paragraphs, and returns i or j depending on which paragraph should come first in the final sorted output. (Don't worry about ties.) This function runs in $O(1)$ time, since almost any two paragraphs can be compared by looking at just their first few characters!
   - MoveParagraph(i, j) 'cuts' out the ith paragraph and 'pastes' it back in as the jth paragraph. This function runs in $O(n_i)$ time, where n_i is the number of characters in the ith paragraph. (So in particular, $n_1 + n_2 + \cdots + n_p = n$.)

     Here is an example of MoveParagraph$(7, 2)$:

Congress shall make no law respecting ...		Congress shall make no law respecting ...
No soldier shall, in time of peace ...		**A well regulated militia, being** ...
The right of the people to be secure ...		No soldier shall, in time of peace ...
No person shall be held to answer for ...		The right of the people to be secure ...
In all criminal prosecutions, the ...		No person shall be held to answer for ...
In suits at common law, where the ...	$\Longrightarrow$	In all criminal prosecutions, the ...
A well regulated militia, being ...		In suits at common law, where the ...
Excessive bail shall not be required ...		Excessive bail shall not be required ...
The enumeration in the Constitution ...		The enumeration in the Constitution ...
The powers not delegated to the ...		The powers not delegated to the ...

   [Hint: For full credit, your algorithm should run in $o(n \log n)$ time when $p = o(n)$.]

4. (continued)

5. Describe and analyze an algorithm to randomly shuffle an array of n items, so that each of the $n!$ possible permutations is equally likely. Assume that you have a function $\textsc{Random}(i, j)$ that returns a random integer from the set $\{i, i+1, \ldots, j\}$ in constant time.

   [Hint: As a sanity check, you might want to confirm that for $n = 3$, all six permutations have probability $1/6$. For full credit, your algorithm must run in $\Theta(n)$ time. A correct algorithm that runs in $\Theta(n \log n)$ time is worth 7 points.]

5. (continued)

From: "Josh Pepper" <jwpepper@uiuc.edu>
To: "Chris Neihengen" <neihenge@uiuc.edu>
Subject: FW: proof
Date: Fri, 29 Sep 2000 09:34:56 -0500

thought you might like this.

Problem: To prove that computer science 373 is indeed the work of Satan.

Proof: First, let us assume that everything in "Helping Yourself with Numerology", by Helyn Hitchcock, is true.

Second, let us apply divide and conquer to this problem. There are main parts:

1. The name of the course: "Combinatorial Algorithms"
2. The most important individual in the course, the "Recursion Fairy"
3. The number of this course: 373.

We examine these sequentially.

The name of the course. "Combinatorial Algorithms" can actually be expressed as a single integer - 23 - since it has 23 letters.
The most important individual, the Recursion Fairy, can also be expressed as a single integer - 14 - since it has 14 letters. In other words:

    COMBINATORIAL ALGORITHMS = 23
    RECURSION FAIRY = 14

As a side note, a much shorter proof has already been published showing that the Recursion Fairy is Lucifer, and that any class involving the Fairy is from Lucifer, however, that proofs numerological significance is slight.

Now we can move on to an analysis of the number of course, which holds great meaning. The first assumtion we make is that the number of the course, 373, is not actually a base 10 number. We can prove this inductively by making a reasonable guess for the actual base, then finding a new way to express the nature of the course, and if the answer

confirms what we assumed, then we're right. That's the way induction works.

What is a reasonable guess for the base of the course? The answer is trivial, since the basest of all beings is the Recursion Fairy, the base is 14. So a true base 10 representation of 373 (base 14) is 689. So we see:

    373 (base 14) = 689 (base 10)

Now since the nature of the course has absolutely nothing to do with combinatorial algorithms (instead having much to do with the work of the devil), we can subtract from the above result everything having to do with combinatorial algorithms just by subtracting 23. Here we see that:

    689 - 23 = 666

QED.