CS 373: Combinatorial Algorithms, Spring 1999

Midterm 1 (February 23, 1999)

Name:	
Net ID:	Alias:

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.

- **Don't panic!**
- Print your name, netid, and alias in the boxes above, and print your name at the top of every page.
- **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 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.
- 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 nonsense, it might be worth partial credit.
- **Don't panic!**

#	Score	Grader
1		
2		
3		
4		
5		

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 answer. Each correct answer earns you 1 point, but each incorrect answer *costs* you $\frac{1}{2}$ point. (You cannot score below zero.)

☐ What is $\sum_{i=1}^{n} i$?

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

☐ What is the solution of the recurrence $T(n) = T(\sqrt{n}) + n$?

☐ What is the solution of the recurrence $T(n) = T(n-1) + \lg n$?

☐ What is the solution of the recurrence $T(n) = 2T\left(\left\lceil \frac{n+27}{2} \right\rceil\right) + 5n - 7\sqrt{\lg n} + \frac{1999}{n}$?

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

☐ 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?

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

☐ How many bits are there in the binary representation of the nth Fibonacci number?

☐ What is the worst-case cost of merging two arbitrary splay trees with n items total into a single splay tree with n items.

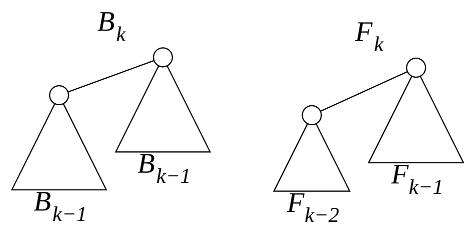
☐ Suppose you correctly identify three of the answers to this question as obviously wrong. If you pick one of the two remaining answers at random, what is your expected score for this problem?

2. (a) **[5 pt]** Recall that a binomial tree of order k, denoted B_k, is defined recursively as follows. B_0 is a single node. For any $k > 0$, B_k consists of two copies of B_{k-1} linked together.

   Prove that the degree of any node in a binomial tree is equal to its height.

   (b) **[5 pt]** Recall that a Fibonacci tree of order k, denoted F_k, is defined recursively as follows. F_1 and F_2 are both single nodes. For any $k > 2$, F_k consists of an F_{k-2} linked to an F_{k-1}.

   Prove that for any node v in a Fibonacci tree, $\text{height}(v) = \lceil \text{degree}(v)/2 \rceil$.



Recursive definitions of binomial trees and Fibonacci trees.

3. Consider the following randomized algorithm for computing the smallest element in an array.

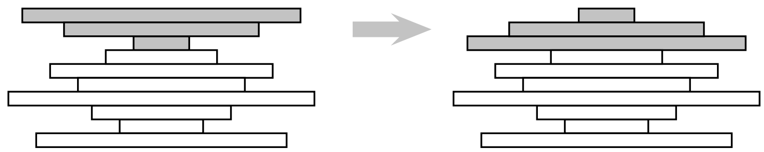
```
RandomMin(A[1 .. n]):
  min ← ∞
  for i ← 1 to n in random order
      if A[i] < min
          min ← A[i]    (⋆)
      return min
```

(a) **[1 pt]** In the worst case, how many times does RandomMin execute line $(\star)$?

(b) **[3 pt]** What is the probability that line $(\star)$ is executed during the nth iteration of the for loop?

(c) **[6 pt]** What is the exact expected number of executions of line $(\star)$? (A correct $\Theta()$ bound is worth 4 points.)

4. Suppose we have a stack of n pancakes of different sizes. We want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation we can perform is a *flip* — insert a spatula under the top k pancakes, for some k between 1 and n, and flip them all over.



Flipping the top three pancakes

   (a) **[3 pt]** Describe an algorithm to sort an arbitrary stack of n pancakes.

   (b) **[3 pt]** Prove that your algorithm is correct.

   (c) **[2 pt]** Exactly how many flips does your algorithm perform in the worst case? (A correct $\Theta()$ bound is worth one point.)

   (d) **[2 pt]** Suppose one side of each pancake is burned. Exactly how many flips do you need to sort the pancakes *and* have the burned side of every pancake on the bottom? (A correct $\Theta()$ bound is worth one point.)

5. You are given an array $A[1\,..\,n]$ of integers. Describe and analyze an algorithm that finds the largest sum of of elements in a contiguous subarray $A[i\,..\,j]$.

   For example, if the array contains the numbers $(-6, 12, -7, 0, 14, -7, 5)$, then the largest sum is $19 = 12 - 7 + 0 + 14$.

   $$\begin{array}{|c|}\hline -6 \\ \hline\end{array}\underbrace{\begin{array}{|c|c|c|c|}\hline 12 & -7 & 0 & 14 \\ \hline\end{array}}_{19}\begin{array}{|c|c|}\hline -7 & 5 \\ \hline\end{array}$$

   To get full credit, your algorithm must run in $\Theta(n)$ time — there are at least three different ways to do this. An algorithm that runs in $\Theta(n^2)$ time is worth 7 points.