CS 373: Combinatorial Algorithms, Summer 2000 IMCS

Midterm 1 (June 17, 2000)

Name:	
Net ID:	

This is a closed-book, no-calculator exam!

You may use an $8\frac{1}{2}'' \times 11''$ sheet of notes (both sides) which you must hand in with your exam.

- You have 75 minutes to complete the exam
- Print your name and netid in the boxes above, and print your name at the top of every page.
- Answer all the questions. They are ten (10) points each.
- 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.
- Don't spend too much time on any single problem. If you get stuck, move on to something else and come back later.
- If you need more than the front of the page, please mark clearly that the answer is continued on the back of that page. If you feel like you want to erase something, don't. Just cross it out and plainly mark your official answer.

#	Score
1	
2	
3	
4	
5	
Total	

1. **Multiple Choice**

   Every question below has one of the following answers.

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

   You need to choose the *best* answer. 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; incorrect answers will not penalize you.

☐ What is $\sum_{k \geq 0} 2^{-k}$?

☐ What is $\sum_{k=1}^{n} k \log k$?

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

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

☐ What is the solution of the recurrence $T(n) = 4T\left(\left\lceil \frac{n-35}{2} \right\rceil\right) + \frac{n}{50} - 7\sqrt[3]{\lg n \lg \lg n} + \frac{1729}{n}$?

☐ The amortized time for inserting one item into an n-node scapegoat tree is $O(\log n)$. What is the average-case time for a sequence of n insertions into an initially empty scapegoat 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 digits are there in the decimal representation of $n \cdot 10^n$?

☐ What is the average-case cost of merging two Fibonacci heaps each with n^2 items?

2. Fast queue with stacks

   (a) Implement a queue (i.e. the two operations Enqueue and Dequeue) using two stacks. You may only use the operations Push and Pop in your implementation, no pointer manipulation.

   (b) Justify the correctness of your implementation.

   (c) Show that the amortized costs of the two queue operations are both $O(1)$ (assuming $O(1)$ costs for the stack operations).

3. Give a $\Theta(n)$ procedure that takes as input an array $A[1..n]$ and performs a random permutation on the array elements. Hint: swap elements randomly by position. In your proof of correctness, you must show that any permutation is equally likely in your algorithm.

4. A Lesson in Business

You have finally accomplished your lifelong goal of owning a data structures store, where customers pay by the operation to use your Fibonacci heap. Unfortunately, the previous owner mismanaged his finances; although your Fibonacci heap consists of a large number of trees and marked nodes, you have no reserve of money (i.e. the potential function Φ is 0).

(a) Show that you can run the store without increasing your prices asymptotically and recover your losses in the next m operations, provided $m = \Omega(n)$.

(b) Show that the condition $m = \Omega(n)$ is necessary, i.e. that the claim in part (a) is not true without it.

5. You're hiking through the woods when you stub your toe on something in the ground. You dig up a pot of gold coins (this happens to be near the beach where you found the treasure chest of gold bricks). Since you've stubbed your toe so many times, you want to go buy some new shoes, and you realize you could use the coins. The coins are all the same size, but they have a number stamped on each one, showing its value. You notice that there are only k different denominations of coins, but a large amount of all of them (note that the k different denominations are arbitrary).

   Being a thoughtful and efficient person, you want to take the fewest number of coins possible to pay for the price n for shoes.

   In other words, you have a target value n, and you want to find the fewest number of coins necessary (if possible) whose value adds up to n.

   Describe and analyze a dynamic programming algorithm to solve the coin changing problem in $O(nk)$ time (i.e. give a recurrence, prove that it gives the optimum.