University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2009

Name:
Email:

- "Nothing is true. All is permitted" - Friedrich Nietzsche. Well, not exactly. *You are not permitted to discuss this exam with any other person.* If you do so, you will surely be smitten. You may consult any *other* sources including books, papers, web pages, computational devices, animal entrails, seraphim, cherubim, etc. in your quest for truth and solutions. Please acknowledge your sources.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer. A numerical solution obtained via a computer program is unlikely to get much credit, if any, without a correct mathematical derivation.
- Write your solution in the space provided for the corresponding problem.
- If any question is unclear, ask for clarification.

Question	Points	Score	Grader
1	20		
2	10		
3	20		
4	20		
5	20		
6	20		
Total	110		

1. **Divide and Conquer**

   - (10 points) Consider the recurrence $f(0) = 2$, $f(1) = 2$, $f(n) = \prod_{i=0}^{n-1} f(i)$. Prove via induction that the solution to this recurrence is $\Omega(2^{2^{n-1}})$

- (10 points) Imagine we change the domino problem we discussed in class as follows. As before, we have a 2 by n rectangle and we want to count the number of ways we can tile it with dominos that are of dimension 2 by 1. However, now there are two colors of dominos: red and black. A red domino must 1) be placed vertically; and 2) be adjacent to a vertically placed black domino. There are no constraints on the black dominos. How many ways can we now tile the rectangle? Show your work.[1] Give an exact solution.

[1]Two tilings are identical if both the colors and orientation of dominos is identical in both tilings.

2. **Poker**

Recall that a standard deck of 52 cards contains one card from each of 13 different ranks: 2 through 10, Jack, Queen, King and Ace; and 4 different suits: hearts, diamonds, clubs and spades. In the game of poker, a *royal flush* consists of 5 cards: 10, Jack, Queen, King, and Ace that are all of the same suit. Thus, there are 4 possible royal flushes that can be formed from a deck of 52 cards. The probability that a randomly dealt hand of 5 cards is a royal flush is thus $4/\binom{52}{5}$.

(a) (8 points) If a dealer turns over n cards, for some $5 \leq n \leq 52$, from a well-shuffled deck of 52 cards, what is the expected number of royal flushes that can be formed from those n cards? Hints: 1) Use linearity of expectation; 2) Check that your answer makes sense for boundary values of n.

(b) (2 points) How many cards need to be turned over before you would expect to be able to form at least 1 royal flush? Hint: Google has a built in calculator that does some combinatorics - try it out by googling “x choose y” for integers x and y.

3. **Expected and Amortized Time**

(a) (10 points) What if you're given a graph where the weight of each edge is distributed uniformly at random between ℓ and u for some real numbers ℓ and u, and you want to find a minimum spanning tree? Describe an efficient *deterministic* algorithm to solve this problem. Please show correctness of your algorithm and give the expected run time (since the input is randomized, the run time will be a random variable). Your algorithm should only use data structures and algorithms discussed in class e.g. you will get no credit for citing e.g. the MST algorithm of Bernard Chazelle that uses soft heaps (soft heaps are inefficient in practice anyway). Note: The run time of your algorithm should be better than $O(n \log n)$ in expectation, since you're now solving just a special case of the MST problem.

(b) Recall from the homework the problem where you need to create a data structure that supports the following operations: Insert(x) and Delete-Top-Half(). For this problem, it was possible to achieve an amortized complexity of $O(1)$ for each operation. In this problem, you will consider cases where the delete operation deletes different numbers of elements. In both cases, assume your data structure must be comparison based i.e. numbers are compared via $\leq$, $<$, $>$, $\geq$ only).

i. (5 points) Imagine that your data structure must support the following operations: Insert(x) and Delete-Max(). The Insert function inserts one element as before, but the Delete-Max function deletes and returns the largest element in your data structure. What is the best amortized complexity you can now get for these operations? Give both upper and lower bounds and justify your answer.

ii. (5 points) Now Imagine your data structure must support the following operations: Insert(x) and Delete-Deciles(d). The Insert function is as before, but the Delete-Deciles(d) function deletes and returns all elements greater than or equal to the $d/10 * n$-th largest element, where n is the number of items in the data structure and d is an integer between 1 and 9. What is the best amortized complexity you can now get for these operations? As before give both upper and lower bounds and justify your answer.

4. ***Summy!***

In this problem, you will be solving a number game called *Summy*. Summy is illustrated by the example below, where the input to the game is the left table and the output is the right table. The game consists of a 3 by 3 table, along with specified values $r_1, r_2, r_3, c_1, c_2, c_3$. In the output, each cell of the table must be filled in with a top and bottom number so as to satisfy the following rules: 1) the two numbers in each cell must sum to 9; 2) The top numbers in each cell in row i must sum to R_i; 3) the bottom numbers in column i must sum to the value C_i.

			16
			13
			11
12	12	17	

3/6	8/1	5/4	16
4/5	5/4	4/5	13
8/1	2/7	1/8	11
12	12	17	

(14 points) Give an efficient algorithm, based on network flow, to find a solution to a game of Summy if a solution exists.

(6 points) Summy has become so popular, that on an international flight you find to your delight that the magazine in your seat pocket has a summy game. Your excitement is short-lived however since the last passenger has filled in some of the cells of this game in ink. Describe an algorithm that can take a summy game, along with a subset of partially filled in cells and return a full solution if one exists. Assume as input, you are given values $r_1, r_2, r_3, c_1, c_2, c_3$ and for some subset of the cells, top and bottom values for each cell in that subset.

5. ***MOD***

You are back on the game show circuit, this time on the 70's British game show *Mod*. You are now playing the following game against an opponent. At the beginning of each round, you and your opponent secretly choose an integer between 0 and 2. At the end of the round, you both reveal your numbers simultaneously and you win $(x + y)$ mod c dollars where x is your number and y is your opponents number and c will be either 2 or 4 depending on the game variant. You goal is to determine a probability distribution over your 3 choices that will maximize your winnings, *given that your opponent knows your strategy and plays optimally for that strategy*! In other words, you want to choose probabilities p_i for each number i such that your winnings are maximized even if your opponent knows your p_i values.

(a) (10 points) Assume $c = 4$. Write down a linear program to find your optimal strategy and expected winning from that strategy. Hint: Use a trick similar to the one that we used in designing the linear program for the shortest paths problem.

(b) (10 points) Now suppose that $c = 5$ and you know that your opponent has an irrational fear of the number 0, and so will never pick it. Write down a linear program to find your optimal strategy and expected winning from that strategy.

6. **Just Business**

Assume there are n goods $g_1, g_2, ..., g_n$ and there is an n by n table T such that one unit of good g_i buys $T[i, j]$ units of good g_j. In this problem you will start with one unit of good g_1 and you want to come up with a sequence of exchanges, $g_1, g_{i2}, ..., g_{ik}, g_1$, that maximizes *profit*, which is the value $T[g_1, g_{i2}] * T[g_{i2}, g_{i3}] * ... * T[g_{ik}, g_1]$.

(a) (10 points) Your boss is concerned about regulations on the "goods" exchange market (What are these goods anyway? What kind of job have you gotten yourself into?), and so asks you if you can find a solution that maximizes profit but never exchanges for an intermediate good more than once. In other words, you start and end with good g_1, but all other goods are distinct. Call this problem UNIQUE-GOODS and show it is NP-Hard by a reduction from an NP-Hard problem we have discussed in class.

(b) (10 points) Your boss is upset you can't solve the previous problem, but is confident you will solve the following one, because he knows you would not like to disappoint the "family" a second time. Your new goal is to find an exchange of goods of length no more than L that maximizes profit. You can now exchange for intermediate goods more than once but the total number of times goods are exchanged can never be more than L. Give an efficient algorithm to solve this problem.