University of New Mexico
Department of Computer Science

Midterm Examination

CS 561 Data Structures and Algorithms
Fall, 2009

Name:	
Email:	

- Print your name and email, *neatly* in the space provided above; print your name at the upper right corner of *every* page. Please print legibly.
- You are permitted to use your class notes, my lecture notes, your hws, the class hw solutions, the textbook and a calculator for this exam. *No other resources are allowed. You are not permitted to discuss this exam with anyone. The exam is "closed internet".*
- Do all problems in this booklet. *Show your work!* You will not get partial credit if we cannot figure out how you arrived at your answer.
- Write your entire answer in the space provided for the corresponding problem. Do not include any additional sheets of paper.
- If any question is unclear, ask us for clarification.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Recurrences and Asymptotics**

   Remember that when the base case for a recurrence is not explicitly given, assume that it is constant for inputs of constant size.

   - Consider the recurrence $f(0) = 1$, $f(n) = \sum_{i=0}^{n-1} f(i)$. Prove that the solution to this recurrence is $\Theta(2^n)$

   - Solve the following recurrence: $f(n) = 4f(n-1) - 4f(n-2) + 2^n$. Do not solve for the constant coefficients

- Solve the following recurrence: $f(n) = f(n/2) + f(n/4)$. Do not solve for the constant coefficients. If an algorithms runtime is given by this recurrence, how would it compare with algorithms with runtimes of $\theta(2^n)$, $\theta(n)$, $\theta(\sqrt{n})$?

2. **Heaps and Sorting**

Professor Humbert claims to have invented a new Max-Heapify function for heaps that runs in $O(\lg \lg n)$ time. Moreover, Humbert claims that his new Max-Heapify function is completely comparison based, i.e. the only way it ever compares two keys is with the $\leq$ operator. Could Humbert be right?

Prove that the following algorithm correctly sorts a list of n elements by induction on n. Don't forget to include the base case, inductive hypothesis and inductive step.

```
SillySort(A, i, j){
  Let n = j - i + 1;                    // n is the number of elements to sort
  If n <= 1 then return;           //nothing to sort
  If n == 2                               // only two elements to sort
    then
      if A[i] > A[j], swap A[i] and A[j]
  else                                      // more than two elements to sort
     SillySort(A,i+n/2,j);         //SillySort the last n/2 elements of A
     SillySort(A,i,i+n/2);    //SillySort the first n/2+1 elements of A
     SillySort(A,i+1,j)            //SillySort the last n-1 elements of A
 }
```

2. **Heaps and Sorting, continued.**

3. **Data Structures**

- Imagine that in a red black tree, the rule that a red node must have two black children is relaxed to the rule that a red node must have at least one black child. All other rules remain the same. What is now the maximum asymptotic height that a red black tree with n nodes can have? Give an example or proof as necessary to justify your claim.

- Imagine you have a "blacklist" of n messages that are spam. When a new message arrives in your mail queue, you want to test it to see if it's a message in your black list. However, space is at a premium and so you don't want to have to store all n messages in memory (assume each message is many bits in length). Answer the following questions with asymptotic notation. Hint: Assume that you have a hash function that maps strings of any size to integers that are essentially random.
  - What is the minimum number of bits of space to ensure that you can test if new messages are in the blacklist and get false positive with probability no more than $1/100$? What is the time needed to test if a message is in the blacklist in this case?
  - How many bits do you need if you want the probability of false positives to be no more than $1/n$? What is the time to test membership in this case?

4. **Probability**

- *Bad Santa II: Mr. Kringle's Revenge:* Consider the Bad Santa problem from hw1 with the following change. The child is allowed to take 2 passes over the sequence of boxes if necessary, but still needs to find just one present before stopping. However, Santa can secretly hide the presents again after the first pass of the child. Describe and analyze an algorithm for this new problem that minimizes the expected number of boxes opened. Hint: The problem is now much easier than the hw problem.

- *Closest Points:* Imagine n points are distributed uniformly at random on a circle with circumference 1. Show that the expected number of pairs of points that are within distance $\theta(1/n^{2)}$ of each other is greater than 1 (note that this implies that the smallest distance between two points is likely $O(1/n^2)$). Hint: Partition the circle into n^2/k regions of size k/n^2 for some constant k; then use the Birthday paradox to solve for the necessary k.

5. **Are you Smarter than a 561 Student?**

   You are competing in the popular game show "Let's Make a Dynamic Program" with another player. You and your opponent both start with 0 dollars. If you reach (or exceed) n dollars before your opponent, you win n dollars; if your opponent reaches (or exceeds) n dollars before you, you win nothing; and if you both reach (or exceed) n dollars at the same time, you both win n dollars. In each turn, you get to choose the level of difficulty of the next question asked, where this difficulty is represented by an integer k between 1 and n. If you answer the question correctly, you get k dollars, otherwise your opponent gets k dollars. Note that you are always in control throughout the entire game of the difficulty level of the question asked.

   Through careful study of the game you have been able to determine the probability p_i for i between 1 and n, which is the probability that you will answer a question of difficulty i correctly.

   - Consider the greedy algorithm where you always choose a question of difficulty level i for i maximizing $i * p_i - i * (1 - p_i) = i(2p_i - 1)$. Is this an optimal algorithm? Hint: Is it ever better to make a long shot bet because the probability of success from multiple short bets is small. In particular,think about the case where your opponent has $n - 1$ dollars and you have 0.

- Let state (i, j) be the state where you have i dollars and your opponent has j dollars. Note that if you choose the difficulty level to be k at that state, you have probability p_k of going to state $(i + k, j)$ and probability $(1 - p_k)$ of going to state $(i, k + jj)$. Now let $e(i, j)$ be your expected winnings if you have i dollars and your opponent has j dollars and you play optimally. Write a recurrence relation for the value $e(i, j)$. Note: You will find it useful to consider i and j values that range from 0 to $2n - 1$. Hint: Use expected values for simpler subproblems and the probabilities p_i described above to compute $e(i, j)$. Don't forget the base case(s).

- Give the pseudocode for computing the value $e(0, 0)$, which gives you your expected winnings if you play this game optimally.

- What if the game is changed as follows. You still select the difficulty level k, but after your selection, both you and your opponent have the chance to write down the answer to the question. Whoever gets the answer correct wins k dollars (note that both of you may win now). There is no penalty for a wrong answer. The probability that you answer a question of difficulty k correctly is p_k and the probability that your opponent answers correctly is q_k. Can you still solve this new problem using dynamic programming? If so, give a recurrence and describe how to change the algorithm. If not, describe why not.