University of New Mexico
Department of Computer Science

Midterm Examination

CS 561 Data Structures and Algorithms
Fall, 2012

Name:	
Email:	

- This exam lasts 75 minutes. It is open book and open notes but no electronic devices are permitted.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer.
- 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	20		
3	20		
4	20		
5	20		
Total	100		

1. **Short Answer**

   Answer the following questions using simplest possible θ notation. Draw a box around your final answer. No need to justify answers for problems on this page.

   (a) Solution to the following recurrence relation: $f(n) = 3f(n/2) + n$.

   (b) Solution to the following recurrence relation: $f(n) = 5f(n-1) - 6f(n-2) + n$.

   (c) Number of balls that must be thrown uniformly and independently into n bins before we expect at least one pair of balls to fall in the same bin.

   (d) Runtime of fastest algorithm to solve the Fractional Knapsack problem over n when the value per pound of each item is distributed independently and uniformly at random between 0 and 1.

(e) Consider the recurrence $f(1) = 2$, $f(n) = f(n/2) * f(3n/4)$. Prove via induction that $f(n) \geq 2^n$. Don't forget the base case (BC), inductive hypothesis (IH) and inductive step (IS). Also make it clear where you are using the IH in the IS.

2. **Amortized Analysis**

Recall that in class we discussed an INCREMENT algorithm for incrementing a binary counter in $O(1)$ amortized time. Now suppose we want to also support a RESET operation that sets all the bits in the counter to 0. Below are the algorithms for INCREMENT and RESET. They make use of an array B of bits and a variable m which is the most significant bit.

Algorithm 1 INCREMENT($B[0, \ldots, \infty], m$)

```
i ← 0
while B[i] = 1 do
    B[i] ← 0
    i ← i + 1
end while
B[i] ← 1
if i > m then
    m ← i
end if
```

Algorithm 2 RESET($B[0, \ldots, \infty], m$)

```
for i ← 0 to m do
    B[i] ← 0
end for
```

In the following questions, let n be the number of operations on this binary counter.

(a) What is the worst-case run time of INCREMENT as a function of n?

(b) What is the worst-case run time of RESET as a function of n?

(c) Prove that in an arbitrary sequence of calls to INCREMENT and RESET, each operation has an amortized cost of O(1).

2. **Amortized Analysis, continued.**

3. **Probability**

Imagine we place m sensor nodes independently and uniformly at random in the cells of a n by n array. We say that a pair of sensors *conflict* if they are in the same row or in the same column.

(a) What is the expected number of pairs of sensors that conflict?

(b) How large must m be for the expected number of conflicting pairs to be greater than 1. Give your answer in $\theta()$ notation as a function of n.

4. **Donuts Revisted**

Assume there are three types of donut boxes: a box containing $x_1 = 1$ donut costing c_1 dollars; a box containing x_2 donuts costing c_2 dollars; and a box containing x_3 donuts costing c_3 dollars. You want to buy exactly n donuts at minimum cost. (Note that since $x_1 = 1$, you can now always buy exactly n donuts for any positive n.)

(a) Let $m(i)$ be the minimum cost of exactly i donuts. Write a recurrence relation for $m(i)$. Don't forget the base case(s).

(b) Now describe in 1-3 sentences how you would create an algorithm to compute the minimum cost way to buy exactly n donuts. We only want a high level description, not the entire algorithm. You will loose points on this answer if you write more than 3 sentences.

(c) Now describe in 3-4 sentences how you would amend your algorithm so that it also returns the set of boxes that achieves the minimum cost. You will loose points if you write more than 4 sentences.

5. **Drunken Debutants**

Recall that in the homework, we had n debutants (debs) randomly assigning themselves to n porsches. Here we consider a new variant. Assume that both the debs and porsches are labelled 1 through n, where porsche i belongs to debutante i. Now the debs leave the party in order from 1 to n. Deb 1 is drunk and enters a porsche uniformly at random. Then for $i = 2, 3, \ldots n$, deb i first checks porsche i. If it's empty, she stays there, if it's full, she enters a porsche chosen uniformly at random from all empty porsches.

In this problem, you will calculate the value $f(n)$, the probability that the last deb enters her own porsche.

(a) Note that $f(1) = 1$. What is $f(2)$?

(b) Write a recurrence relation for $f(n)$. Hint 1: Deb 1 enters porsche 1 with probability $1/n$. What is $f(n)$ in this case? How about if deb 1 enters porsche 2? In that case, can you write $f(n)$ as $f(j)$ for some $j < n$. Hint 2: The right hand side of your recurrence should contain many $f(j)$ values.

(c) Now use the guess and check method to solve for the exact value of $f(n)$ for $n \geq 2$. Don't forget to include base case, IH, and IS.
Hint: To get a good guess, you may want to compute $f(3)$ and $f(4)$ from the recurrence in order to spot a trend.