15-251 : Great Theoretical Ideas In Computer Science

Fall 2013

Assignment 6

$\boxed{{}^{63.546}_{29}\mathbf{Cu}}$tting $\boxed{{}^{40.078}_{20}\mathbf{Ca}}$ke

Due: Thursday, Oct. 17, 2013 11:59 PM

Name: ________________________________

Andrew ID: ________________________________

Question:	1	2	3	4	5	Total
Points:	30	25	25	20	10	110
Score:						

1. Crazy handful of big-O

A 15-251 professor by the name of Allan has recently been diagnosed with terminal bipartite manifolds. To support his family when he is gone, he decides to manufacture a material known as "cake", which students have been using to improve their performance in classes. He teams up with Jessica, an old student of his, and they bake their first cake together. They only make a few thousand off of it, and because butter is so scarce, they change recipes. They replace the butter with soybean oil, which they will steal from a nearby warehouse. The warehouse has an unobtainium lock that can't be broken or melted. Instead, they will figure out the code by solving the math problems inscribed in the lock:

Consider four positive functions from $\mathbb{R}$ to $\mathbb{R}$, $f(x)$, $g(x)$, $u(x)$, and $v(x)$ where $f(x) \in O(g(x))$ and $u(x) \in O(v(x))$. Prove or disprove the following: [You need to do both part (e) and part (f) separately.]

(5) (a) $f(x) + u(x) \in O(g(x) + v(x))$

(5) (b) $f(x) * u(x) \in O(g(x) * v(x))$

(5) (c) $f(x)^{u(x)} \in O(g(x)^{v(x)})$

(5) (d) $f(u(x)) \in O(g(v(x)))$

(5) (e) $\frac{1}{n} \in O(\frac{1}{n^2})$

(5) (f) $\frac{1}{n} \in \Omega(\frac{1}{n^2})$

2. **Breakage**
Allan bakes his best cake of all time. It comes out a little blue, but it is 99% delicious. He needs to divide the cake among his n customers. In lecture we proved that if n is a power of two, there is an algorithm that takes $O(n \log n)$ queries to find a Proportional Allocation. Sadly, Allan lives in the real world, where n might not be a power of two.

(25) (a) Design an algorithm that always finds a Proportional Allocation for any n, and prove that it uses $O(n \log n)$ queries.

3. **Kafcakesque**

Jessica has to divide a cake amongst herself and two friends, Ferret and Skippy. The three of them just finished an OS assignment, so they haven't eaten in three weeks. If any person a values b's piece more than his own, things will end poorly.

Consider the following algorithm for cutting the cake, where there are three people, Agent 1, Agent 2, and Agent 3:

1. Agent 1 divides the cake into three equally-valued pieces X_1, X_2, X_3 such that $V_1(X_1) = V_1(X_2) = V_1(X_3) = \frac{1}{3}$. Without loss of generality, assume $V_2(X_1) \geq V_2(X_2) \geq V_2(X_3)$.
2. Agent 2 trims the largest piece according to V_2 to create a tie for largest, that is, agent 2 removes $X' \subseteq X_1$ such that $V_2(X_1 \setminus X') = V_2(X_2)$. We call the three pieces — one of which is trimmed — cake 1 ($X_1 \setminus X'$, X_2, X_3), and we call the trimmings cake 2 (X').

   Division of cake 1:
3. Agent 3 chooses one of the three pieces of cake 1.
4. If agent 3 chose the trimmed piece ($X_1 \setminus X'$), agent 2 chooses between the two other pieces of cake 1. Otherwise, agent 2 receives the trimmed piece. We denote the agent $i \in \{2, 3\}$ that received the trimmed piece by T , and the other agent by $\bar{T}$.
5. Agent 1 receives the remaining piece of cake 1.

   Division of cake 2:
6. Agent $\bar{T}$ divides cake 2 into three equally-valued pieces.
7. Agents $T, 1, \bar{T}$ select a piece of cake 2 each, in that order.

Jessica is too exhausted to figure out whether this algorithm actually works — she needs your help.

(10) (a) Prove that the allocation of cake 1 is envy free.

(15) (b) Prove that the allocation of the entire cake is envy free. [Hint: The allocation of cake 2, on its own, may not be envy free.]

4. **Greenest Car Detergent (Greatest Common Denominator)**
Allan's wife, Mrs. Poe, has to launder the profits from Allan's business. She buys a carwash, and runs the cash through the cleaning equipment. She is worried about any government investigation, so she decides to appease the EPA by finding the Greenest Car Detergent that she can find. A car detergent is just a cleaning product of prime ingredients.

For this problem, please make the following assumptions:
1. Both the following algorithms terminate, and are correct. 2. All operations on integers are constant-time (independent of sizes of the integers). 3. All input integers are represented in binary.

Mrs. Poe comes up with the following algorithm to determine the GCD of two detergents:

```
(*computes the GCD of m and n, where m > n*)
fun gcd1(m:int,n:int):int=
  if n=0 then
    m
  else
    gcd1(n,m mod n)
```

(10) (a) Let m_i and n_i be the values of m and n respectively in the i^{th} recursive call to gcd1. Prove that $n_{i+2} \leq n_i/2$, if at least $i+2$ calls are made to gcd1. (Do not step through the code like you did in 15-150. There is no need to be this pedantic.)

(5) (b) Prove that the worst-case running time of gcd1 is (at most) polynomial in the input size, that is, the running time is $O(g(\text{input-size}))$ for some polynomial g.

Mrs. Poe is having trouble implementing the mod operation, so she decides to design an algorithm that doesn't use it:

```
(*computes the GCD of m and n *)
fun gcd2(m:int,n:int):int=
  if n>m then
    gcd2(n-m,m)
  else if m>n
    gcd2(m-n,n)
  else
    m
```

(5) (c) Prove that the worst-case running time of gcd2 is (at least) exponential in the input size, that is, the running time is $\Omega(c^{\text{input-size}})$ for some constant $c > 1$.

5. **Felina (Bonus)**

Allan needs to come up with a schedule for baking a cake. For a given recipe, there are n tasks to complete. Let's say there are k time slots. A schedule is an assignment of tasks to time slots. Between any two tasks a and b, there may be a conflict. If there is a conflict, the two tasks cannot be performed at the same time. There are no other restrictions on the time slots of the tasks. Allan wants to know how many possible schedules there are.

Let $R = (T, C)$ be a recipe-graph where T is the set of tasks (vertices) and C is the set of conflicts (edges). That is, $\{a, b\} \in C$ if and only if a and b cannot be performed in the same time slot. Let $f_R(k)$ be the number of possible schedules (valid assignments of times slots to tasks) for the recipe R with k available time slots. Note that R is simply a graph.

(3) (a) If R is a cycle of size n, determine $f_R(k)$.

(3) (b) Now, let R be any recipe-graph. Let e be a conflict between two tasks. Let $R - e$ denote the recipe where that conflict is removed. Let R/e denote the recipe where the two tasks involved in conflict e are combined into a single task, and thus are forced to occur in the same time slot. Prove that $f_R(k) = f_{R-e}(k) - f_{R/e}(k)$.

(4) (c) Again, let R be any recipe-graph. Prove the function $f_R(k)$ is a polynomial in k of degree at most n, where n is the number of tasks in R.