CSE332 Data Abstractions, Spring 2010
Homework 1

Due: **Friday, April 9, 2010** at the beginning of class. Your work should be readable as well as correct.

This assignment has **six** problems.

Problem 1. Some important sums

A certain set of sums appear in this course, and more importantly in the real world, repeatedly in analyzing the running time of different algorithms. In this problem, you will compute two of these sums and prove a third to be true.

1. Weiss 1.8a
2. Weiss 1.8b
3. Weiss 1.12a

(For problems 1.8a and 1.8b make sure to not only evaluate the sum, but also show how you performed this evaluation.)

Problem 2. Horner's Rule

The classic way to evaluate a polynomial is called Horner's rule which can be stated recursively as follows. Let $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. To compute $p(c)$ for some constant c, first evaluate $q(c)$ where $q(x) = a_1 + a_2x + \cdots + a_nx^{n-1}$ recursively, then $p(c) = a_0 + cq(c)$.

1. Provide a base case for this method. That is, explain how to do the "last step" without recursion.
2. Prove, by induction, that Horner's method, including your base case, works for any n.
3. For a polynomial of degree n, as a function of n, how many additions and how many multiplications are used to evaluate the polynomial in Horner's rule.
4. Provide an elegant, **non-recursive** pseudocode function for Horner's rule where the coefficients are stored in an array A, with A[i] containing a_i. Hint: this can be done in about 5 lines of code.

Problem 3. Using a Time Budget

This problem gives an orthogonal view of comparative running times from that given in lecture. Be sure to look at the patterns in your table when you have completed it. For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds. For large entries (say, those that warrant scientific notation), an estimate is sufficient. For one of the rows, you will not be able to solve it analytically, and will need a calculator, spreadsheet, or small program.

$f(n)$	1 second	1 minute	1 hour	1 day	1 month	1 year
$1000 \log_2 n$						
$100n$						
$100n \log_2 n$						
$10n^2$						
n^3						
$\frac{1}{10}2^n$						

Problem 4. Fun with Induction

The following statement is clearly not true. Can you spot the error in the inductive "proof" below? Specify which of the following 5 numbered lines are wrong, and clearly describe the error.

> **All jelly beans are the same color**
>
> **"Proof"**: The proof is by induction on n:
>
> Base case ($n = 1$):
>
> 1. If there is only one jelly bean in the set, then the statement trivially holds.
>
> Induction step: ($n = k+1$). Assume the statement holds for $n = k$. Now suppose you have $k+1$ jelly beans.
>
> 2. Set the first one aside. The remaining k must be the same color (let's say red).
> 3. All we have to do now is show that the first one is also red.
> 4. To do this, remove a second jelly bean and put the first jelly bean back in to form a new set of size k. By the inductive hypothesis, all the jelly beans in the new set are also the same color.
> 5. Since this set contains $k-1$ jelly beans that we already know are red, it follows that they are all red (including the first).

Problem 5. Big-O, Big-Θ

Big-O, Big-Θ, and Big-Ω are the ubiquitous language of the analysis of algorithms. Getting your head around what these notations mean is essential for understanding pretty much any theoretical analysis of an algorithm.

Prove true or explain why the following statements are incorrect:

1. If $f(n) = O(g(n))$ and $h(n) = O(k(n))$, then $f(n) - h(n) = O(g(n) - k(n))$.
2. If $f(n) = O(g(n))$ and $h(n) = O(k(n))$, then $f(n) + h(n) = O(g(n) + k(n))$.
3. $\left(2^{n+3}\right) = \Theta(2^n)$
4. $(2^n)^{1/3} = \Theta(2^n)$

Problem 6. Algorithm analysis

1. Weiss 2.7a (give the best big-O bound you can for each of the 6 program fragments)
2. Weiss 2.11