# CSE 326 DATA STRUCTURES
# HOMEWORK 1

Due: **Friday, January 15, 2010** at the beginning of class. Your work should be readable as well as correct.

**Problem 1. Big-$O$, Big-$\Theta$**

Big-$O$, Big-$\Theta$, and Big-$\Omega$ are the ubiquitous language of the analysis of algorithms. Getting you 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:

(a) If $f(n) = O(g(n))$ and $h(n) = O(k(n))$, then $f(n) - h(n) = O(g(n) - k(n))$.

(b) If $f(n) = O(g(n))$ and $h(n) = O(k(n))$, then $f(n) + h(n) = O(g(n) + k(n))$.

(c) $(2^n)^{1/3} = \Theta(2^n)$

(d) $\left(2^{n+3}\right) = \Theta(2^n)$

**Problem 2. Some important sums**

A certain set of sums appear in this course, and more importantly in the real world, over and over again in analyzing the running time of different algorithms. In this problem, you will do a few of these sums and prove one to be true. (For (a) and (b), be sure to not just evaluate the sum, but show us how you performed this evaluation.)

(a) $\sum_{i=0}^{\infty} \frac{1}{4^i}$

(b) $\sum_{i=0}^{\infty} \frac{i}{4^i}$

(c) $\sum_{i=1}^{N} (2i-1) = N^2$ (Prove this is true)

**Problem 3. Horner's Rule**

The classic way to evaluate a polynomial is called Horners 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 $p(c) = a_0 + cq(c)$.

(a) Provide a base case for this method, and prove, by induction, that the method works for any $n$.

(b) 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.

(c) Provide an elegant, **non-recursive** pseudocode function for Horner's rule where the data coefficients are stored in an array A[0 . . . n], with A[i] containing $a_i$. Hint: this can be done in $\sim 5$ lines of code.

**Problem 4. Fun with Induction**

The following statement is clearly not true. Can you spot the major 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. Algorithm analysis**

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

**Problem 6. Fibonacci**

Consider a recursive function that computes the Fibonacci series (see also Weiss 2.4.2). In this problem you'll prove that it runs in exponential time. The number of operations is described by the following recurrence relation (assume n=0 corresponds to the first number in the series).

$T(0) = T(1) = 2$ `/* cost of test, return */`

$T(n) = T(n-1) + T(n-2) + 3$ `/* cost of test, addition, two recursive calls, return */`

Prove by induction that $T(n) > (\frac{3}{2})^n$