CS350—Winter 2014
Homework 1

Due Thursday 16th January 2014, on paper, in class. This assignment will be graded. You can alternatively turn in your work on d2l, if you so wish. Assignments up to 1 week late will be docked 20%. Notes on presenting a proof are here.

1. For each of the following six functions, state its rate of growth using Θ notation; if possible, use one of the Basic Asymptotic Efficiency Classes from Levitin Table 2.2. Explain your reasoning in one line. Then sort the functions from lowest to highest order of growth.

   a. $(n - 1)!$
   b. $n!$
   c. $n^3 / 1000 + 100\, n^2 + 500000$
   d. $\sqrt{(64\, n)}$
   e. 2^{2n}
   f. $\log_{10} n^6$
   g. 2^{n+1}

2. Prove (by using the definitions of the notations involved) that if $t(n) \in O(g(n))$, then $g(n) \in \Omega(t(n))$.

3. $p(n) = a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} + a_{k-3} n^{k-3} + \ldots + a_0$, where the a_i are constants, is a polynomial of degree k. Prove that every polynomial of degree k belongs to $\Theta(n^k)$

4. Levitin mentions, in section 2, that one can check whether all elements of an array are distinct by a two-part algorithm that first sorts the array, and then scans through the array looking at adjacent elements.

   If the sorting is done by an algorithm with the time efficiency in $\Theta(n \log n)$, what will be the time efficiency class of the entire algorithm? Explain why.

5. Door in a wall. (Optional; for extra credit) You are facing a wall that stretches infinitely in both directions. There is a door in the wall, but you know neither how far away nor in which direction. You can see the door only when you are right next to it. [Based on a problem from Ian Parberry's *Problems on Algorithms*, 1995].

   Design an algorithm that enables you to reach the door by walking at most $O(n)$ steps where n is the (unknown to you) number of steps between your initial position and the door.

   Either:

   - write an expression for the number of steps that your algorithm will take, and show that it is indeed in $O(n)$, or
   - write a program that simulates your algorithm, counts the number of steps that you take, and tabulates the results for a range of values of n.