Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2004
Homework 1b
Due on Tuesday, 09/07/04 (beginning of class)

1. [**10 points**] **The Geometric Series.**

   (a) Use induction on n to show that for all integers $n \geq 0$

   $$1 + a + a^2 + a^3 + \ldots + a^n = \frac{a^{n+1} - 1}{a - 1}$$

   where a is some arbitrary real number other than 1.

   (b) Explain where your induction proof relied on the fact that $a \neq 1$.

   (c) What does this expression evaluate to when $a = 1$?

2. [**10 points**] State whether each of the following statements is true or false and then carefully **prove** your answer using the formal definition of Big-Oh notation (the first five) or Big-Omega notation (the last one) and properties of logarithms from Homework 1a. Remember that to show that one of these statements is false, you must obtain a formal contradiction; it does not suffice to just say "false".

   (a) $2^{n+1} = O(2^n)$.

   (b) $2^{2n} = O(2^n)$.

   (c) $3n^2 \log_2 n + 16n = O(n^3)$.

   (d) $25 \log_2 8n^{10} = O(\log_{10} n)$.

   (e) $4^{\log_2 n} = O(n^3)$.

   (f) $\frac{n}{2} \log_2(\frac{n}{2}) = \Omega(n \log_2 n)$.

3. [**15 Points**] **Ranking Functions!** List the functions below in increasing order by placing one function on each line, with the top line containing the smallest function and the bottom line containing the largest function. If two functions are in the same group (functions $f(n)$ and $g(n)$ are in the same group if $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$ then write those two functions together on the same line. (You should assume, that the size of the problem, n, is an integer in all cases.) In this problem, we are not asking for proofs (that would take a long time!), just the correct ranking.

$\sqrt{n}$	n^2	$(\sqrt{2})^{\log_2 n}$	$(\frac{3}{2})^n$
$4^{\log_2 n}$	$n \log_{10} n$	$n^{1/3} + \log_2 n$	1.0001^n
2^n	42	$\log_2 n^2$	$\log_2 4^n$
e^n	$\log_2 n!$	$(\log_2 n)^2$	n^e

4. **[15 points]**

   Consider sorting n numbers by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then find the second smallest element of A and exchange it with $A[2]$. Continue in this manner for the first $n-1$ elements of A. This algorithm is known as *selection sort*.

   - Write pseudocode for this algorithm. You can use the book's conventions or code similar to that given in class (syntax isn't important, as long as it's clear).
   - Describe in a sentence the loop variant this algorithm maintains.
   - Demonstrate that this algorithm is correct by proving your loop invariant via induction.
   - Does this algorithm always terminate and if so, why?
   - Why does selection sort only need to loop over the first n-1 elements?

5. **[25 Points] The Stock Market Problem!** You've been hired as Director of Algorithm Design by the brokerage firm of Weil, Proffet, and Howe. Periodically, the firm looks at how a particular stock has done in the last n days and wishes to know what the maximum possible profit would have been if someone had bought and sold a share during those n days. More formally, let A be an array of n numbers where $A[i]$ represents the value of a share in the stock on day i. We wish to find

   $$\max_{1 \leq j < k \leq n} A[k] - A[j]$$

   The interpretation of this expression is that on day j we could buy a share for $A[j]$ dollars and at a later date, k, we could sell the share for $A[k]$ dollars, making a profit of $A[k] - A[j]$ dollars. It's critical, of course, that $j < k$ since one can only sell a share *after* they have purchased it!

   (a) Describe a simple algorithm for solving this problem (that is, determining the maximum profit) and show that this algorithm has time complexity $\Theta(n^2)$.

   (b) Describe a recursive divide-and-conquer algorithm for this problem.

   (c) Write the recurrence relation for the running time of your divide-and-conquer algorithm and explain how you got it.

   (d) Use the recursion tree method (see the example of Merge-sort that we did in class) to find a closed-form formula for the running time of your algorithm and express this in Θ notation. Show each step of your analysis very carefully. How does this algorithm compare to the simple $\Theta(n^2)$ algorithm?