Algorithms
Computer Science 140 & Mathematics 168
Spring 2009
Homework 1b
Due Tuesday, January 27

- Please use LaTeX to typeset your solutions to this assignment.
- Please remember to submit each problem on a **separate sheet of paper** with your name on the top. (If a problem requires multiple sheets, please staple them together).
- About bonus problems: Please at least read the bonus problems when they are given. If you have the time and inclination, work on them! Working on these problems will give you a deeper understanding of the material. Moreover, the extra points on these problems are potentially substantial. Since these are equivalent to regular homework points, you may use the points to opt out of regular homework problems on other problem sets or just boost your course grade up!

1. [**10 points**] Indicate whether each of the following statements is true or false and then carefully **prove** your answer using the formal definition of Big-Oh notation 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) $2^{n+1} \in O(2^n)$.

   (b) $2^{2n} \in O(2^n)$.

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

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

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

2. [**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. To help you establish the ranking, you will need to compare pairs of functions and you can use any methods that you want. Some useful methods are algebraic manipulation (e.g. rewriting a function in another form that allows you to compare it more easily with a second function), l'Hôpital's rule to compute the limit of the ratio of pairs of functions, and numerical experiments (e.g. choose n to be some large power of 2 and then evaluate the two functions on that value).

$\sqrt{n}$	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

3. **[15 Points] Proving the Correctness of Insertion Sort.** In class we stated invariants that can be used to prove the correctness of Insertion Sort. Use these (or your own) invariants to prove the correctness of Insertion Sort. Make sure that your proof is complete and self-contained. State the invariants that you are using and give a complete proof of correctness. Below is the pseudo-code for Insertion Sort from class:

```
InsertionSort(A){
   for i = 2 to n
   {
      current = i
      while (current > 1 AND A[current-1] > A[current]){
         swap elements in locations current-1 and current
         current = current - 1
      }
   }
}
```

4. **[25 Points] Stock Ratings!** Algorithm design is an active area of work in the investment industry. You've been hired as the Director of Algorithm Design by the brokerage firm of Weil, Proffet, and Howe.

The firm specializes in rating stocks. Their measure of the quality of a stock is the longest consecutive number of days in which the stock did not decrease in value. For example, consider the stock values below:

Day:	1	2	3	4	5	6	7	8
Value:	42	40	45	45	44	43	50	49

In this example, the length of the longest consecutive non-decreasing run is 3. This run goes from day 2 to day 4. (In some cases there could be ties for the longest run, but that's OK! We are only looking for the length of the longest run, not the actual run itself.)

Here are your tasks:

(a) First, describe a very simple "naive" algorithm for this problem (an explanation in English suffices) and explain briefly why the worst-case running time of this algorithm is $\Theta(n^2)$ where n is the length of the array of stock values.

(b) Next, use the divide-and-conquer paradigm to devise an algorithm whose running time is asymptotically better than $\Theta(n^2)$. Provide pseudo-code (the syntax is up to you, as long as it is clear to any computationally literate reader) or a very clear English description of your algorithm. (*Note: There are algorithms that are fast and do not use divide-and-conquer. Please use divide-and-conquer here though! Moreover, there are several different divide-and-conquer algorithms for this problem. Some are asymptotically faster than n^2 and some are asymptotically MUCH faster than n^2. Any algorithm asymptotically faster than n^2 suffices here, but if you find an algorithm **and** can prove that your algorithm is asymptotically optimal, then you will receive 5 extra bonus points. It's not too hard! Try it!)*

(c) Assuming that n is a power of 2, give a recurrence relation that describes the worst-case running time of your algorithm.

(d) Solve your recurrence relation to get a simple closed form for the running time of your algorithm and then give the asymptotic (Θ notation) running time. Show your solution in detail. (*Note:* Do not appeal the the Master Theorem in the algorithms books to give you the solution to the recurrence relation. You should solve

the recurrence relation yourself using the work tree method that we used in class.) If you want to use a picture to help explain the running time, you may just leave some space in your document and draw it by hand or embed a figure that you've drawn in another tool. Download the source for this document from the course website and read the embedded comments on how to do all this in LaTeX.

(e) Explain briefly why the asymptotic running time still holds even if n is not a power of 2.

(f) Prove the correctness of your algorithm using mathematical induction.

5. **[25 Point OPTIONAL BONUS PROBLEM] Chip Testing!** *This is a fun and interesting problem! It is highly recommended!*

Professor Chip Testor has n supposedly identical computer chips that in principle are capable of testing each other. The professor's test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the answer of a bad chip cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Chip A says	Chip B says	Conclusion
B is good	A is good	Both good, or both bad
B is good	A is bad	At least one is bad
B is bad	A is good	At least one is bad
B is bad	A is bad	At least one is bad

(a) Show that if more than $\frac{n}{2}$ chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

(b) Consider the problem of finding a single good chip from among n chips, assuming that more than $\frac{n}{2}$ of the chips are good. Show that $\lfloor \frac{n}{2} \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

(c) Show that the good chips can be identified with $\Theta(n)$ pairwise tests, assuming that more than $\frac{n}{2}$ of the chips are good. Explain clearly why your algorithm identifies all of the good chips. Then give and solve the recurrence that describes the number of tests.