Prof. Sergio A. Alvarez
21 Campanella Way, room 569
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CS383, Algorithms
Spring 2009
HW1

1. Arrange the functions defined below in order of their asymptotic growth rates, from smallest to largest. Include the sorted list, and explanations that justify your answer. Logarithms are in base 2. $n!$ denotes the factorial of n (product of all integers between 1 and n, inclusive).

   (a) n^n
   (b) $n \log n$
   (c) n^2
   (d) 2^n
   (e) $n^{\log n}$
   (f) $n!$
   (g) $(\log n)^n$

2. The first two Fibonacci numbers are $F_0 = 0$ and $F_1 = 1$. Subsequent Fibonacci numbers satisfy the following recurrence relation:

$$F_n = F_{n-1} + F_{n-2}$$

   (a) Formally describe the computational task of computing the n-th Fibonacci number F_n, by precisely stating what the inputs and desired outputs are (and how the outputs should be related to the inputs). Note any restrictions on the input and output values.
   (b) Translate the above recursive definition of the Fibonacci numbers into an algorithm (in pseudocode) for computing F_n. The algorithm should follow the given recurrence relation as closely as possible.
   (c) How efficient is the algorithm described in the preceding subtask? Does it run in polynomial time? Exponential time? Justify your answer.
   (d) (Optional) Can you describe a more efficient algorithm for the computation of F_n? Explain in detail.

3. Consider the sequence X_n defined by the base cases and recurrence relation below:

$$X_1 = 1, \quad X_2 = 2, \quad X_n = \frac{4}{3}X_{n-1} + \frac{1}{3}X_{n-2} \text{ if } n > 2$$

In this task you will examine the growth rate of X_n as a function of n.

(a) First some apparent good news. Use mathematical induction (clearly stating the basis, induction hypothesis, and inductive step; see my notes on induction) to prove that

$$X_n \leq 2^n \text{ for all } n \geq 0$$

Thus, the sequence X_n grows slower than the exponential function 2^n.

(b) Now prove the sobering fact that the growth rate of X_n is exponential nonetheless:

$$X_n \geq 1.4^n \text{ for all } n \geq 4$$

Again use induction. *Hint:* 1.4^2 *is less than 2.*

(c) By the above, we know that X_n is between 1.4^n and 2^n. Could we be so lucky that X_n is actually *equal* to b^n for some yet-to-be found base b? Assuming that this is the case, use the recurrence relation defining X_n to find an equation satisfied by b. Can you solve this equation to find the value of b? Explain in detail.

4. In this task you will capture the asymptotic growth rate of $\log(n!)$, where $n!$ denotes the factorial of n.

(a) Explain why $n! < n^n$.

(b) Is it true that $n! > (n/2)^n$? Justify your answer.

(c) Using the upper bound for $n!$ above, and a suitable modification of the exponent in the would-be lower bound, find a simple big Θ expression for $\log(n!)$. Explain.

5. (Optional programming task)

(a) Implement the following (heuristic) primality test based on Fermat's little theorem as a function in either C++ or Java:

```
function isFermatPrime(n)
pick a random integer a between 1 and n-1
if a^(n-1) is congruent to 1 (mod n), return true;
otherwise return false
```

Submit your source code.

(b) Use your implementation above, together with a second function that correctly decides primality, to write a program that determines the error rate of the random Fermat test among d-digit numbers, for each d between 1 and 7. Submit your documented source code, as well as the output of your program.

(c) Modify the Fermat primality tester function so that it uses multiple test integers (a) instead of one. How does the error rate depend on the number of test integers?