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 Solutions

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) $(\log n)^n$

   Solution

   Asymptotically, the right order is:

   $$n \log n < n^2 < n^{\log n} < 2^n < (\log n)^n < n! < n^n$$

   Justifications follow.

   - $n \log n = O(n^2)$ because $\log n = O(n)$ (multiply both sides by n).
   - $n^2 = O(n^{\log n})$ since both sides have the same base but the exponent $\log n$ on the right is greater than the exponent 2 on the left for all values of n greater than $2^2 = 4$.
   - $n^{\log n} = O(2^n)$ by taking log on both sides: the left side becomes $(\log n)^2$, while the right becomes n, which is asymptotically larger than $(\log n)^2$ by L'Hôpital's rule (see my notes on asymptotic running time).
   - $2^n = O((\log n)^n)$ by taking log on both sides also: the left side becomes n, while the right becomes $n \log \log n$ (where $\log \log n$ means the composite function $\log(\log n)$).
   - $(\log n)^n = O(n!)$ by task 4, since $\log \log n = O(\log n)$ by the substitution $\log n = u$.
   - Finally, $n! = O(n^n)$ since $n! \leq n^n$ by direct comparison of the n factors on both sides: $n! = 1 * 2 * 3 * \cdots n \leq n * n * n * \cdots n = n^n$.

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.

Solution

Inputs: An integer $n \geq 0$.

Output: The n-th Fibonacci number F_n.

(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.

Solution

The pseudocode for the algorithm follows.

```
function recursiveFibonacci(n)
        if n=0 return 0
        if n=1 return 1
        return recursiveFibonacci(n-1) + recursiveFibonacci(n-2)
```

(c) How efficient is the algorithm described in the preceding subtask? Does it run in polynomial time? Exponential time? Justify your answer.

Solution

The running time of the above algorithm is *exponential* in the input size, n. This is easy to say but harder to actually understand. To see that $F(n)$ (shorthand) requires exponential time, examine the recursion tree for that invocation. The number of nodes in the tree provides a lower bound for the running time, since each node represents a function that call that takes at least one step. The recursion tree for $T(n)$ consists of a root node from which two subtrees hang: one of the subtrees is the recursion tree for $T(n-1)$ and the other is the recursion tree for $T(n-2)$. It follows that the number of nodes in the whole tree is at least as large as F_n, the n-th Fibonacci number. Why is F_n exponential in n? Argue along the lines of task 3 (exercise).

(d) (Optional) Can you describe a more efficient algorithm for the computation of F_n? Explain in detail.

Solution

My suggestion is to proceed by analogy with the more efficient combinatorial coefficient calculator that we discussed in class, using a suitable table or array and a judicious calculation order that avoids the redundancy inherent in straightforward reliance on the recursive definition. I'll let you work out the details (they're in the book, too).

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.

Solution

- (Basis) $X_1 = 1 < 2 = 2^1$ and $X_2 = 2 < 4 = 2^2$.
- (Induction Hypothesis) $X_k \leq 2^k$ for all values of $k \leq n$.
- (Inductive Step) Assuming that the induction hypothesis holds, we will show that $X_{n+1} \leq 2^{n+1}$. Start from the definition of X_{n+1} in terms of the previous two terms:

$$X_{n+1} = \frac{4}{3}X_n + \frac{1}{3}X_{n-1}$$

By the induction hypothesis, $X_n \leq 2^n$ and $X_{n-1} \leq 2^{n-1}$. Adding these bounds, we get

$$X_{n+1} \leq \frac{4}{3}2^n + \frac{1}{3}2^{n-1}$$

Since $2^n = 22^{n-1}$, the sum of the terms on the right equals 32^{n-1}, while 2^{n+1} equals 42^{n-1}. Hence,

$$X_{n+1} \leq 3 * 2^{n-1} < 2^{n+1}$$

This is what we wanted to prove. We're done!

(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.*

Solution

- (Basis) $X_4 = 14/3 > 4 = 2^2 \geq (1.4^2)^2 = 1.4^4$ because $2 \geq 1.4^2$.
- (Induction Hypothesis) $X_k \geq 1.4^k$ for all values of k between 4 and some n that is no smaller than 6.
- (Inductive Step) Assuming that the induction hypothesis holds, we show that $X_{n+1} \geq 1.4^{n+1}$. From the definition of X_{n+1}:

$$X_{n+1} = \frac{4}{3}X_n + \frac{1}{3}X_{n-1}$$

while by the induction hypothesis we have the bounds: $X_n \geq 1.4^n$ and $X_{n-1} \geq 1.4^{n-1}$. Add the bounds to get:

$$X_{n+1} \geq \frac{4}{3}1.4^n + \frac{1}{3}1.4^{n-1} \geq 2 * 1.4^{n-1} \geq 1.4^2 * 1.4^{n-1} = 1.4^{n+1},$$

where we again used the fact that $2 \geq 1.4^2$. This completes the proof.

(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.

Solution

As stated, let's *assume* that $X_n = b^n$ for some unknown b. Using the recurrence relation that defines X_n, we then have:

$$b^n = \frac{4}{3}b^{n-1} + \frac{1}{3}b^{n-2}$$

This is true for all $n \geq 3$. Pick $n = 3$ for simplicity (things work just fine for larger values, too, but this is easiest). We get the following quadratic equation for b:

$$b^2 = \frac{4}{3}b + \frac{1}{3}1$$

This equation is solvable! Solving, we find two roots:

$$b = \frac{1}{3}\left(2 \pm \sqrt{7}\right)$$

Which one should we pick? We can't really pick either. You can check that using either value b_1 or b_2, the powers b_i^n indeed satisfy the given recurrence equation. Since the recurrence is linear, any linear combination $A * b_1^n + B * b_2^n$ of the two power sequences also satisfies the recurrence! The solution is to pick the right coefficients A and B. Can you do this and finish up this line of reasoning?

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

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

Solution

We did this in the first task, above.

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

Solution

Frustratingly, this is not true. Values as small as $n = 6$ fail, and the degree to which the bound fails increases as n grows.

(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.

Solution

The desired lower bound is $n! > (n/2)^{n/2}$. This follows by examining the factors in the product that defines $n!$: the last $n/2$ or so terms are larger than $n/2$, and the rest are larger than 1. Combining the upper and lower bounds, and taking logs, we get the following inequalities for large enough values of n:

$$n/2 \log n/2 < \log n! < n \log n$$

The $\log n/2$ term is the same as $\log n - 1$ (since $\log 2 = 1$). This shows that $\log n!$ is sandwiched between two terms that are both $\Theta(n \log n)$. Hence, $\log n = \Theta(n \log n)$.

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?