Homework 1

Wednesday, April 13, 2016 8:50 PM

Matthew Koken

3) Complexity of fibonacci sequence.

Both recursive and iterative will need to perform the same number of calculations. Therefore, we can look at the fibonacci sequence in general. (assuming no dynamic programming, which will do less additions overall)

So:

$f_0 = 0$

$f_1 = 1$

$f_n = f_{n-1} + f_{n-2} < 2^n$, $n \in$ natural numbers

$2^n < n!$

Cases:

$n = 0$

$f_0 = 0 < 2^0$

$0 < 1$ ✓

$n = 1$

$f_1 = 1 < 2^1$

$1 < 2$ ✓

$n = 2$

$f_2 = f_1 + f_0 < 2^2$

$1 + 0 < 4$

$1 < 4$ ✓

$n = k + 1$

$f_{k+1} = f_k + f_{k-1} < 2^k + 2^{k-1}$

$f_k < 2^k$

$f_{k+1} < 2^{k+1}$

$f_k + f_{k-1} < 2^k(1 + 1/2)$

$(1 + 1/2) < 2$

$2^k(1 + 1/2) < 2^k(2)$

so $f_k + f_{k-1} < 2^k(2)$

and $f_k + f_{k-1} < 2^{k+1}$

$\therefore$

so $O(2^n)$ order of additions.

4. Levitin, 2.1: 5, 7, 8, 9
5. Levitin, 2.2: 2, 3

4) 2.1:

5. a. Prove formula (2.1) for the number of bits in the binary representation of a positive decimal integer.

b. Prove the alternative formula for the number of bits in the binary representation of a positive integer n:

$$b = \lceil \log_2(n + 1) \rceil.$$

c. What would be the analogous formulas for the number of decimal digits?

d. Explain why, within the accepted analysis framework, it does not matter whether we use binary or decimal digits in measuring n's size.

a) $b = \lfloor \log_2 n \rfloor + 1$

$$\underbrace{10\ldots0}_{b-1} \Rightarrow \text{smallest pos int. } 2^{b-1}$$

$$\underbrace{11\ldots1}_{b-1} \Rightarrow 2^{b-1} + 2^{b-2} + \ldots + 1 = 2^b - 1$$

$$2^{b-1} \leq n < 2^b$$

$$\log_2 2^{b-1} \leq \log_2 n < \log_2 2^b$$

$$b-1 \leq \log_2 n < b$$

$$b-1 = \lfloor \log_2 n \rfloor$$

$$b = \lfloor \log_2 n \rfloor + 1$$

b) $b = \lceil \log_2 (n+1) \rceil$

Same smallest pos.

$$2^b \leq n < 2^{b+1}$$

$$\log_2 2^b \leq \log_2 n < \log_2 2^{b+1}$$

$$b \leq \log_2 n < b+1$$

$$b \leq \log_2 n+1 < b$$

$$b = \lceil \log_2 (n+1) \rceil$$

c) $B = \lfloor \log_{10} n \rfloor + 1$

d) The size of n will correspond to the base of the log in the same way, so number size will not matter.

7. Gaussian elimination, the classic algorithm for solving systems of n linear equations in n unknowns, requires about $\frac{1}{3}n^3$ multiplications, which is the algorithm's basic operation.

a. How much longer should you expect Gaussian elimination to work on a system of 1000 equations versus a system of 500 equations?

b. You are considering buying a computer that is 1000 times faster than the one you currently have. By what factor will the faster computer increase the sizes of systems solvable in the same amount of time as on the old computer?

1000/500 = 2
so 2n

a) $\frac{T(2n)}{T(n)} \approx \frac{C_m \frac{1}{3}(2n)^3}{C_m \frac{1}{3} n^3} = \boxed{8}$

b) $T_{old}(n) \approx C_m \frac{1}{3} n^3$

$T_{new}(n) \approx 10^{-3} C_m \frac{1}{3} N^3$

$T_{old}(n) = T_{new}(N) \Rightarrow C_m \frac{1}{3} n^3 \approx 10^{-3} C_m \frac{1}{3} N^3$

so $\frac{N}{n} \approx \boxed{10}$

8. For each of the following functions, indicate how much the function's value will change if its argument is increased fourfold.

a. $\log_2 n$ **b.** $\sqrt{n}$ **c.** n **d.** n^2 **e.** n^3 **f.** 2^n

a) $\log_2 4n - \log_2 n = (\log_2 4 + \log_2 n) - \log_2 n = \boxed{2}$

b) $\frac{\sqrt{4n}}{\sqrt{n}} = \frac{4\cancel{n}}{\cancel{n}} = \boxed{4}$

c) $\frac{4n}{n} = \boxed{4}$

d) $\frac{(4n)^2}{n^2} = \frac{16n^2}{n^2} = \boxed{16}$

e) $\frac{(4n)^3}{n^3} = \frac{64n^3}{n^3} = \boxed{64}$

f) $\frac{2^{4n}}{2^n} = 2^{3n} = \boxed{(2^n)^3}$

9. For each of the following pairs of functions, indicate whether the first function of each of the following pairs has a lower, same, or higher order of growth (to within a constant multiple) than the second function.

a. $n(n+1)$ and $2000n^2$ **b.** $100n^2$ and $0.01n^3$
c. $\log_2 n$ and $\ln n$ **d.** $\log_2^2 n$ and $\log_2 n^2$
e. 2^{n-1} and 2^n **f.** $(n-1)!$ and $n!$

a) $n(n+1)$ and $2000n^2$

$n^2 + n$ both are n^2 $2000n^2$

and within a constant

b) $100n^2$ and $0.01n^3$

n^2 n^3

quadratic $<$ cubic

c) $\log_2 n$ and $\ln n$

the base can be changed so it doesn't matter

$\log_2 n$ and $\ln n$

are both $\log_a$ so same level of growth

d) $\log_2^2 n$ and $\log_2 n^2$

$\log_2 n \log_2 n$ $2\log_2 n$

so $\log_2^2 n$ has a higher growth rate than $\log_2 n^2$

e) 2^{n-1} and 2^n

$\frac{1}{2}2^n$

so 2^{n-1} and 2^n have the same order of growth

f) $(n-1)!$ and $n!$

$n! = (n-1)!\,n$

$(n-1)!$ has a lower growth than $n!$

5

2.2: 2, 3

2. Use the informal definitions of O, Θ, and Ω to determine whether the following assertions are true or false.

TABLE 2.2 Basic asymptotic efficiency classes

Class	Name	Comments
1	*constant*	Short of best-case efficiencies, very few reasonable examples can be given since an algorithm's running time typically goes to infinity when its input size grows infinitely large.
$\log n$	*logarithmic*	Typically, a result of cutting a problem's size by a constant factor on each iteration of the algorithm (see Section 4.4). Note that a logarithmic algorithm cannot take into account all its input or even a fixed fraction of it: any algorithm that does so will have at least linear running time.
n	*linear*	Algorithms that scan a list of size n (e.g., sequential search) belong to this class.
$n \log n$	*linearithmic*	Many divide-and-conquer algorithms (see Chapter 5), including mergesort and quicksort in the average case, fall into this category.
n^2	*quadratic*	Typically, characterizes efficiency of algorithms with two embedded loops (see the next section). Elementary sorting algorithms and certain operations on $n \times n$ matrices are standard examples.
n^3	*cubic*	Typically, characterizes efficiency of algorithms with three embedded loops (see the next section). Several nontrivial algorithms from linear algebra fall into this class.
2^n	*exponential*	Typical for algorithms that generate all subsets of an n-element set. Often, the term "exponential" is used in a broader sense to include this and larger orders of growth as well.
$n!$	*factorial*	Typical for algorithms that generate all permutations of an n-element set.

$n(n+1)/2 \approx n^2/2 \Rightarrow$ quadratic

a. $n(n+1)/2 \in O(n^3)$ **b.** $n(n+1)/2 \in O(n^2)$
c. $n(n+1)/2 \in \Theta(n^3)$ **d.** $n(n+1)/2 \in \Omega(n)$

a) $n(n+1)/2 \in O(n^3)$ true

b) $n(n+1)/2 \in O(n^2)$ true

c) $n(n+1)/2 \in \Theta(n^3)$ false

d) $n(n+1)/2 \in \Omega(n)$ true

3. For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (Use the simplest $g(n)$ possible in your answers.) Prove your assertions.

a. $(n^2+1)^{10}$ **b.** $\sqrt{10n^2+7n+3}$
c. $2n\lg(n+2)^2 + (n+2)^2\lg\frac{n}{2}$ **d.** $2^{n+1}+3^{n-1}$
e. $\lfloor \log_2 n \rfloor$

a) $(n^2+1)^{10} \approx (n^2)^{10} = n^{20} \in \Theta(n^{20})$

$(n^2+1)^{10}$ ... $\lim (n^2+1)^{10}$ ... $\lim (n^2+1)^{10}$ ... $= \lim (1+\frac{1}{\ldots})^{10} = 1$

$\lim_{n\to\infty} \frac{\quad}{n^{20}} = \lim_{n\to\infty} \frac{\quad}{(n^2)^{10}} = \lim_{n\to\infty} \left(\frac{\quad}{n^2} \right) = \lim_{n\to\infty} \left(\quad \frac{\quad}{n^2} \right) = 1$

$\therefore (n^2+1)^{10} \in \Theta(n^{20})$

b) $\sqrt{10n^2 \cancel{+7n} \cancel{+3}} \approx \sqrt{10n^2} = \sqrt{10}n \in \Theta(n)$

$\lim_{n\to\infty} \frac{\sqrt{10n^2+7n+3}}{n} = \lim_{\infty} \sqrt{\frac{10n^2+7n+3}{n^2}} = \lim_{n\to\infty} \sqrt{10 + \frac{7}{n} + \frac{3}{n^2}} = \sqrt{10}$

$\frac{7}{n} \to 0$, $\frac{3}{n^2} \to 0$

so $\sqrt{10n^2+7n+3} \in \Theta(n)$

c) $2n\, lg(n+2)^2 + (n+2)^2 lg\frac{n}{2}$ // $lg = ln$?

$= 4n\,lg(n+2) + (n+2)^2(lg\,n - lg\,2)$

$= 4n\,lg(n+2) + (n+2)^2(lg\,n - 1) \in \Theta(n\,lg\,n) + \Theta(n^2 lg\,n)$

$= \Theta(n^2 lg\,n)$

d) $2^{n+1} + 3^{n-1}$

$= 2^n 2 + 3^n \frac{1}{3} \in \Theta(2^n) + \Theta(3^n)$

$= \Theta(3^n)$

e) $\lfloor \log_2 n \rfloor \approx \log_2 n \in \Theta(\log n)$

$x-1 < \lfloor x \rfloor \leq x$

$\lfloor \log_2 n \rfloor \leq \log_2 n$

$\lfloor \log_2 n \rfloor > \log_2 n - 1 \geq \log_2 n - \frac{1}{2}\log_2 n$, $n \geq 4$

$= \frac{1}{2}\log_2 n \Rightarrow \lfloor \frac{1}{2}\log_2 n \rfloor \approx \log_2 n$

so

$\lfloor \log_2 n \rfloor \in \Theta(\log_2 n) = \Theta(\log n)$