Alan Marcero
Algorithm Analysis

Professor Ed Harcourt

Fri Sept 29th, 2006

<u>Problem #1</u>:
Use Induction to prove the following formula: $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

1) What is the base case?

n == 1

$$\sum_{i=0}^{1-1} 2^0 = 2^1 - 1$$
$$\sum_{i=0}^{0} 1 = 2^1 - 1$$
$$1 = 2^1 - 1$$
$$1 = 1$$

The above summation will run 1 time, thus the summation is equal to 1.

2) What is your induction hypothesis?
We are assuming that the original formula is true: $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

3) What are you trying to prove in your induction step?
Using our assumption that the original formula is true, we are trying to prove that it remains true for k or:

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Proof:

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

We take out one k from the summation and add it to the end to get:

$$\sum_{i=0}^{k} 2^i = \sum_{i=0}^{k-1} 2^i + 2^k = 2^{k+1} - 1$$

This makes the summation equal to the summation in our induction hypothesis:

$$\sum_{i=0}^{k-1} 2^i + 2^k = 2^{k+1} - 1$$

thus:

$$2^k - 1 + 2^k = 2^{k+1} - 1$$

Simplify:
$2(2^k)-1=2^{k+1}-1$

Thus our induction hypothesis is true:
$2^{k+1}-1=2^{k+1}-1$

Problem #2:
Page 69, #10:

```
GE(A[0..n - 1, 0..n])
     //Input: An n-by-n + 1 matrix A[0..n - 1, 0..n] of
real
     //numbers

for i = 0 to n - 2 do
    for j = i + 1 to n - 1 do
        for k = i to n do
             A[j, k] = A[j, k] - A[i, k] * A[j, i] / A[i, i]
```

Derive a sigma (summation) notation of this algorithm and then solve the summation. From this, derive the big-oh of the equation.

$\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\sum_{k=i}^{n}1$ The summation of the above algorithm

$\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}(n-i+1)$

$\sum_{j=i+1}^{n-1}(n-i+1)=\sum_{j=i+1}^{n-1}n-\sum_{j=i+1}^{n-1}i+\sum_{j=i+1}^{n-1}1$

We temporarily drop the left-most summation and factor out the constants:
$n\sum_{j=i+1}^{n-1}1-i\sum_{j=i+1}^{n-1}1+\sum_{j=i+1}^{n-1}1$
Break down each summation into its run time equivalent (top of the summation, minus the bottom, plus the right side):

$n[(n-1)-(i+1)+1]-i[(n-1)-(i+1)+1]+(n-1)-(i+1)+1$

We run into the term (n-1)-(i+1)+1 three times, so let's figure that out first:
$(n-1)-(i+1)+1$
$n-1+i-1+1$
$=n-1-i$

We plug n-1-i back into the original formula for every (n-1)-(i+1) + 1:

$$n(n-1-i)-i(n-1-i)+(n-1-i)$$

Distribute the variables:

$$n^2-n-ni-ni+i+i^2+n-1-i$$

This simplifies to:

$$n^2-2ni+i^2-1$$

We plug this into the left most summation of the original problem:

$$\sum_{i=0}^{n-2}(n^2-2ni+i^2-1)$$

This breaks down into four separate summations:

$$\sum_{i=0}^{n-2}n^2-\sum_{i=0}^{n-2}2ni+\sum_{i=0}^{n-2}i^2-\sum_{i=0}^{n-2}1$$

Factor out the constants:

$$n^2\sum_{i=0}^{n-2}1-2\mathrm{n}\sum_{i=0}^{n-2}i+\sum_{i=0}^{n-2}i^2-\sum_{i=0}^{n-2}1$$

Apply summation formula #2, Page 470, Levitin, to the 2nd summation and simplify:

$$-2\mathrm{n}\sum_{i=0}^{n-2}i=$$

$$-2\mathrm{n}\left[\frac{(n-2)(n-2+1)}{2}\right]$$

$$-2\mathrm{n}\left[\frac{(n-2)(n-1)}{2}\right]$$

$$(n-2)(n-1)$$

$$=n^2-n-2\mathrm{n}+3$$

$$=n^2-3\mathrm{n}+3$$

Put this back into our original fraction and simplify:

$$-2n\left[\frac{n^2-3n+3}{2}\right]$$

$$\frac{-2n}{1}\left[\frac{n^2-3n+3}{2}\right]$$

$$\frac{-2n^3+6n^2-6n}{2}$$

Apply summation formula #3, Page 470, Levitin, to the third summation:

$$\sum_{i=0}^{n-2} i^2 =$$

$$\frac{(n-2)(n-2+1)(2(n-2)+1)}{6}$$

$$\frac{(n-2)(n-1)(2n-4+1)}{6}$$

$$\frac{(n-2)(n-1)(2n-3)}{6}$$

Simplify:

$$(n-2)(n-1)$$

$$=n^2-n-2n+2$$

$$=n^2-3n+2$$

Simplify:

$$n^2-3n+2(2n-3)$$

$$2n^3-3n^2-6n^2+9n+4n-6$$

$$2n^3-9n^2+13n-6$$

Plug this back into our original fraction from above:

$$\frac{2n^3-9n^2+13n-6}{6}$$

Break down the first summation and simplify:

$$n^2\sum_{i=0}^{n-2} 1 = n^2(n-2-0+1)$$

$$=n^2(n-1)$$

$$=n^3-n^2$$

Break down the fourth summation and simplify:

$$\sum_{i=0}^{n-2} 1=(n-2-0+1)$$

$$(n-1)$$

Plug all four broken down summations to get our final formula to simplify:

$$(n^3-n^2)-\frac{-2n^3+6n^2-6n}{2}+\frac{2n^3-9n^2+13n-6}{6}-(n-1)$$

Obtain a common denominator:

$$\frac{n^3-n^2}{1}\cdot\frac{6}{6}=\frac{6n^3-6n^2}{6}$$

$$\frac{(n-1)}{1}\cdot\frac{6}{6}=\frac{6n-6}{6}$$

$$\frac{-2n^3+6n^2-6n}{2}\cdot\frac{3}{3}=-\frac{-6n^3+18n^2-18n}{6}$$

Plug our formulas with common denominators into the original “final formula” to simplify:

$$\frac{6n^3-6n^2}{6}-\frac{-6n^3+18n^2-18n}{6}+\frac{2n^3-9n^2+13n-6}{6}-\frac{6n-6}{6}$$

Subtract:

$$\frac{6n^3-6n^2}{6}-\frac{-6n^3+18n^2-18n}{6}=\frac{12n^3+24n^2-18n}{6}$$

Add:

$$\frac{12n^3+24n^2-18n}{6}+\frac{2n^3-9n^2+13n-6}{6}=\frac{14n^3+15n^2-5n-6}{6}$$

Subtract:

$$\frac{14n^3+15n-5n-6}{6}-\frac{6n-6}{6}=\frac{14n^3+15n^2-5n-12}{6}$$

QED:

$$\frac{14n^3+15n^2-5n-12}{6}$$

Big Oh:

$\frac{14n^3+15n^2-5n-12}{6}$ is Big-Oh n^3

Proof:

$$\frac{14n^3+15n^2-5n-12}{6}\leq C_1 n^3$$

Add 5n and 12 to the left side to get:

$$\frac{14n^3+15n^2}{6}\leq C_1 n^3$$

Replace the n^2 by n^3 to get:

$$\frac{14n^3+15n^3}{6}\leq C_1 n^3$$

$$\frac{29n^3}{6}\leq C_1 n^3$$

Replace the constant value:

$$C_1=\frac{29}{6}$$

$$\frac{29n^3}{6}=\frac{29n^3}{6}$$

QED

Problem #3:
Page 76, #1 (d):

Solve the following recurrence relation:
x(n) = x(n/2) + n for n > 1, x(1) = 1 (solve for $n = 2^k$)

x(1) = 1
x(n) = x(n/2) + n
for $n = 2^k$

x(n)	work
$x(\frac{2^k}{2^0})$	$\frac{n}{2^0}$
$x(\frac{2^k}{2^1})$	$\frac{n}{2^1}$
$x(\frac{2^k}{2^2})$	$\frac{n}{2^2}$
$\vdots$	$\vdots$
$x(\frac{2^k}{2^i})$	$\frac{n}{2^i}$
$\vdots$	$\vdots$
$x(\frac{2^k}{2^{k-1}})$	$\frac{n}{2^{k-1}}$
$x(\frac{2^k}{2^k})$	$\frac{n}{2^k}=1$

$$\sum_{i=0}^{k}\frac{n}{2^i}=n\sum_{i=0}^{k}\left(\frac{1}{2}\right)^i$$

We will now show that ($n = 2^k$) = (log n = k)
$n = 2^k$
log(n) = log(2^k)
log(n) = k log(2) – *(by log law #3, Page 469, Levitin)*
log(n) = k(1) – *$\log_2(2) = 1$ by log law #2, Page 469, Levitin)*
log(n) = k

Thus: $n\sum_{i=0}^{k}\left(\frac{1}{2}\right)^i=n\sum_{i=0}^{\log n}\left(\frac{1}{2}\right)^i$

Using summation formula #5, Page 470, Levitin:

$$n\sum_{i=0}^{\log n}(\frac{1}{2})^i = n\left[\frac{\left(\frac{1}{2}\right)^{\log n+1}-1}{\left(\frac{1}{2}\right)-1}\right]$$

$$= n\left[\frac{\left(\frac{1}{2}\right)^{\log n+1}-1}{-\frac{1}{2}}\right]$$

Multiply the improper fraction by -2 to remove the denominator:

$$-2\text{n}\left[\left(\frac{1}{2}\right)^{\log n+1}-1\right]$$

$\log_2(2) = 1$ by log law #2, Page 469, Levitin:

$$-2\text{n}\left[\left(\frac{1}{2}\right)^{\log n+\log 2}-1\right]$$

$\log_a x + \log_a y = \log_a xy$ by the reverse of log law #4, Page 469, Levitin:

$$-2\text{n}\left[\left(\frac{1}{2}\right)^{\log 2\text{n}}-1\right]$$

$a^{\log_b x} = x^{\log_b a}$ by log law #6, Page 469, Levitin:

$$-2\text{n}\left[(2\text{n})^{\log\frac{1}{2}}-1\right]$$

$\log_a x/y = \log_a x - \log_a y$ by log law #5, Page 469, Levitin:

$$-2\text{n}\left[(2\text{n})^{\log 1-\log 2}-1\right]$$

$\log_a 1 = 0$ and $\log_a a = 1$ by log laws #1 and #2, Page 469, Levitin:

$$-2\text{n}\left[(2\text{n})^{0-1}-1\right]$$

0-1 = -1:

$$=-2n\left[2n^{-1}-1\right]$$

$2n^{-1}$ = 1/2n:

$$-2n\left[\frac{1}{2n}-1\right]$$

Multiply by -2n:

$$\frac{-2n}{1}\cdot\left[\frac{1}{2n}-1\right]=\frac{-2n}{2n}+2n$$

-2n/2n = -1, thus:

$$\frac{-2n}{2n}+2n=2n-1$$

<u>Problem #4</u>:
Page 77, #5 (a):

Given the Towers of Hanoi problem, estimate the number of years it would take to move 64 disks if the monks could move only one disk per minute (Assume that monks do not eat, sleep, or die).

Let M be the moves required to move n disks from peg 1 to peg 3.
M(n) = $2^n - 1$ (proved on pg. 73, Levitin)

For n = 64

M(64) = $2^{64} - 1$
M(64) = 18,446,744,073,709,551,615 moves are required to move 64 disks from peg 1 to peg 3.

The "monks" can only move one disk per minute, thus:
18,446,744,073,709,551,615 * 1 move per minute =
18,446,744,073,709,551,615 minutes are required to move 64 disks from peg 1 to peg 3 at a rate of one disk per minute.

The amount of minutes in a 24 hour day = 60mins * 24hours = 1,440
18,446,744,073,709,551,615 / 1,440 = Approximately 12,810,238,940,076,078 days are required to move 64 disks from peg 1 to peg 3 at a rate of one disk per minute.

The amount of days in 1 year = 365. We divide the amount of days required to move 64 disks at a rate of 1 disk per minute by the amount of days in a year:
12,810,238,940,076,078 / 365 = 35,096,545,041,304 years are required to move 64 disks at a rate of one disk per minute.

Given the Towers of Hanoi problem, approximately <u>35 trillion years</u> are required to move 64 disks at a rate of one disk per minute.

<u>Problem #5</u>:
Page 77, #7, Parts a, c, & d

(a) Design a recursive algorithm for computing 2n for any nonnegative integer n that is based on the formula: $2^n = 2^{n-1} + 2^{n-1}$

```
int pow2(int n) {
     if(n == 0)
          return 1; //the base case
     else
          return pow2(n-1) + pow2(n-1);
}
```

(c) Draw a tree of recursive calls for this algorithm and count the number of calls made by the algorithm.

$$x(n) \qquad\qquad\qquad \text{n work}$$
$$/ \quad \backslash$$
$$x(n-1) + x(n-1)$$
$$/ \quad \backslash \quad / \quad \backslash$$
$$x(n-2) + x(n-2) + x(n-2) + x(n-2)$$
$$\vdots \qquad\qquad \vdots$$
$$x(0)_1 + x(0)_2 + \ldots + x(0)_{2^{n-1}} + x(0)_{2^n}$$

(there are n copies of x(0); each sub-number represents that copy of n)

$$1_1 + 1_2 + \ldots + 1_{2^{n-1}} + 1_{2^n}$$

1 is returned 2^n times (one for each copy of x(0)); each sub-number represents that return of 1. 2^n additions of 1 are equal to 2^n work making this a $O(2^n)$ algorithm.

(d) Is it a good algorithm for solving this problem?
No. Why: Because this recursive implementation runs at an efficiency of 2^n. Whereas implementing this algorithm with a for loop runs at an efficiency of n.