Chapter 0– Solutions

January 30, 2007

0.1. a) $n - 100 = \Theta(n - 200)$
b) $n^{1/2} = O(n^{2/3})$
c) $100n + \log n = \Theta(n + (\log n)^2)$
d) $n \log n = \Theta(10n \log 10n)$
e) $\log 2n = \Theta(\log 3n)$
f) $10 \log n = \Theta(\log(n^2))$
g) $n^{1.01} = \Omega(\log^2 n)$
h) $n^2/\log n = \Omega(n(\log n)^2)$
i) $n^{0.1} = \Omega((\log n)^{10})$
j) $(\log n)^{\log n} = \Omega(n/\log n)$
k) $\sqrt{n} = \Omega((\log n)^3)$
l) $n^{1/2} = O(5^{\log_2 n})$
m) $n2^n = O(3^n)$
n) $2^n = \Theta(2^{n+1})$
o) $n! = \Omega(2^n)$
p) $(\log n)^{\log n} = O(2^{(\log_2 n)^2})$
q) $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$

0.2. By the formula for the sum of a partial geometric series, for $c \neq 1$: $g(n) = \frac{1-c^{n+1}}{1-c} = \frac{c^{n+1}-1}{c-1}$.

a) $1 > 1 - c^{n+1} > 1 - c$. So: $\frac{1}{1-c} > g(n) > 1$.
b) For $c = 1$, $g(n) = 1 + 1 + \cdots + 1 = n + 1$.
c) $c^{n+1} > c^{n+1} - 1 > c^n$. So: $\frac{c}{1-c}c^n > g(n) > \frac{1}{1-c}c^n$.

0.3. a) Base case: $F_6 = 8 \geq 2^{6/2} = 8$.
Inductive Step: for $n \geq 6$, $F_{n+1} = F_n + F_{n-1} \geq 2^{n/2} + 2^{(n-1)/2} = 2^{(n-1)/2}(2^{1/2}+1) \geq 2^{(n-1)/2}2 \geq 2^{(n+1)/2}$.

b-c) The argument above holds as long as we have, in the inductive step, $2^{c(n-1)}(2^c + 1) \geq 2^{c(n+1)}$, i.e. as long as $2^c \leq \frac{1+\sqrt{5}}{2}$.

0.4. a) For any 2×2 matrices X and Y:

$$XY = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix} = \begin{pmatrix} x_{11}y_{11} + x_{12}y_{21} & x_{11}y_{12} + x_{12}y_{22} \\ x_{21}y_{11} + x_{22}y_{21} & x_{21}y_{12} + x_{22}y_{22} \end{pmatrix}$$

This shows that every entry of XY is the addition of two products of the entries of the original matrices. Hence every entry can be computed in 2 multiplications and one addition. The whole matrix can be calculated in 4 multiplications and 4 additions.

b) First, consider the case where $n = 2^k$ for some positive integer k. To compute, X^{2^k}, we can recursively compute $Y = X^{2^{k-1}}$ and then square Y to have $Y^2 = X^{2^k}$ Unfolding the recursion, this can be seen as repeatedly squaring X to obtain $X^2, X^4, \cdots, X^{2^k} = X^n$. At every squaring, we are doubling the exponent of X, so that it must take $k = \log n$ matrix multiplications to produce X^n. This method can be easily generalized to numbers that are not powers of 2, using the following recursion:

$$X^n = \begin{cases} (X^{\lfloor n/2 \rfloor})^2 & \text{if } n \text{ is even} \\ X \cdot (X^{\lfloor n/2 \rfloor})^2 & \text{if } n \text{ is odd} \end{cases}$$

The algorithm still requires $O(\log n)$ matrix multiplications, of which $\log n$ are squares and at most $\log n$ are multiplications by X.

c) The entries of the matrix are the addition of the products of the entries of the original matrix. Addition leaves the number of bits close to unvaried (add at most one bit), while multiplication adds the number of bits of the operands. Hence, every time we square the matrix X, we double the number of bits of its entries. By b), we are squaring X $\log n$ times and hence all intermediate results must be of length less or equal to $2^{\log n} = O(n)$. The multiplications by X leave the sizes of the intermediate result almost unchanged (add at most one bit) and can be neglected, as well as the additions of the products of the entries within the matrix multiplication.

In the next two questions we assume that $M(n) = n^c$ for some $c \geq 1$.

d) The algorithm performs $O(\log n)$ matrix multiplications; each matrix multiplication consists of a constant number of arithmetic operations by a). Each arithmetic operation is on results of size $O(n)$ and hence takes at most time $O(M(n))$. This implies the algorithm runs in time $O(M(n) \log n)$.

e) Let $T(n)$ be the running time of the algorithm on input of size n. In the first level of the recursion, we first run the algorithm for inputs of size $n/2$, which takes time $T(n/2)$. Then, we square the results to obtain the final answer: this last operation takes time $M(n/2)$ as the results of the recursive call have bit size $O(n/2)$ by c). This implies:

$$T(n) = T(n/2) + M(n/2)$$

Expanding the recursion and applying the formula for geometric series, we obtain:

$$\begin{aligned} T(n) &= M(n/2) + M(n/4) + M(n/8) + \cdots + M(1) \leq \\ &\leq \frac{n^c}{2^c} + \frac{n^c}{4^c} + \cdots + \frac{n^c}{8^c} = \\ &= n^c \sum_{i=1}^{\infty} \frac{1}{2^{ic}} = O(n^c) = O(M(n)) \end{aligned}$$