CS 344: Homework #1

Due on September 20, 2017

Professor Bahman Kalantari Section #1

Tina Janulis, Caleb Rodriguez, and Ray Zhang

Problem 1

In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

(a) $f(n) = n - 100$
$g(n) = n - 200$

When $c_1 = 1$, $c_2 = 100$ and $n = 300$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$
Therefore, $f(n) = \Theta(g(n))$

(c) $f(n) = 100n + \log n$
$g(n) = n + (\log n)^2$

When $c_1 = 100$, $c_2 = 200$ and $n = 1$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$
Therefore, $f(n) = \Theta(g(n))$

(e) $f(n) = \log 2n$
$g(n) = \log 3n$

When $c_1 = 1$, $c_2 = 200$ and $n = 3$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$
Therefore, $f(n) = \Theta(g(n))$

(g) $f(n) = n^{1.01}$
$g(n) = n\log^2 n$

When $c = 2$ and $n = 3$, $f(n) \leq cg(n)$
Therefore, $f(n) = \Omega(g(n))$

(i) $f(n) = n^{.01}$
$g(n) = n\log^2 n$

When $c = 2$ and $n = 100$, $f(n) \leq cg(n)$
Therefore, $f(n) = \Omega(g(n))$

(k) $f(n) = n^{.01}$
$g(n) = n\log^2 n$

When $c = 2$ and $n = 30$, $f(n) \leq cg(n)$
Therefore, $f(n) = \Omega(g(n))$

(m) $f(n) = n2^n$
$g(n) = 3^n$

When $c = 200$ and $n = 1$, $f(n) \leq cg(n)$
Therefore, $f(n) = \Omega(g(n))$

(o) $f(n) = n!$
$g(n) = 2^n$

When $c = 1$ and $n = 200$, $f(n) \leq cg(n)$
Therefore, $f(n) = \Omega(g(n))$

(q) $f(n) = \sum_{i=1}^{n} i^k$
$g(n) = n^{k+1}$

$\sum_{i=1}^{n} = n(n^k)$
This could then be translated into $f(n) = n^{k+1}$.
This is equivalent to $g(n) = n^{k+1}$ Therefore, $f(n) = \Theta(g(n))$

Problem 2

Show that, if c is a positive real number, then $g(n) = 1 + c^1 + c^2 + ... + c^n$ is:

(a) $\Theta(1)$ if $c < 1$.

If $0 < c < 1$ and its exponent n is increasing, c^n will decrease as n increases.
Essentially, $\lim_{x \to 0} g(n) = -\infty$.
Therefore, 1 is the leading number and $g(n) = \Theta(1)$.

(b) $\Theta(n)$ if $c = 1$.

If $c = 1$, 1 is being added to itself n times, which translates to
$\sum_{i=1}^{n} 1^n = n$.
Therefore, n is the leading coefficient and $g(n) = \Theta(n)$.

(c) $\Theta(c^n)$ if $c > 1$.

If $c > 1$, c^n will keep increasing until the last c^n which will be the highest number in the series.
Essentially, $\lim_{x \to 0} g(n) = \infty$.
Therefore, c^n is the leading number and $g(n) = \Theta(c^n)$.

Problem 3

The Fibonacci numbers $F_0, F_1, F_2, ...,$ are defined by the rule

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

In this problem we will confirm that this sequence grows exponentially fast and obtain some bounds on its growth.

(a) After considering the function, say $n = 6$. Then, for the basic step we can say:

$F_6 = F_5 + F_4$
$F_6 = 5 + 3$
$F_6 = 8$

With this in mind, we can prove that $F_n \geq 2^{0.5n}$ for $n = 6$:

$$F_6 \geq 2^{0.5(6)}$$
$$8 \geq 2^3$$
$$8 \geq 8 \text{ which is true.}$$

For the inductive step we can first show that when $n = n + 1$:

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

And based on the basic step, we can start proving $F_n \geq 2^{0.5n}$ inductively:

$$\begin{aligned} F_n + F_{n-1} &\geq 2^{0.5(n-1)} + 2^{0.5n} \\ &\geq 2^{0.5(n-1)}(\sqrt{2} + 1) \\ &\geq 2 * 2^{0.5(n-1)} \\ &\geq 2^{0.5n-0.5+1} \\ &\geq 2^{0.5n+0.5} \\ &\geq 2^{0.5(n+1)} \end{aligned}$$

Based on the given equation, $F_n = F_{n-1} + F_{n-2}$, we can conclude that $F_n + F_{n-1} = Fn + 1$.
We are trying to prove that $F_n \geq 2^{0.5n}$.
The inductive proof above can translate to:

$$F_{n+1} \quad \geq \quad 2^{0.5(n+1)}$$

Which proves, inductively, that $F_n \geq 2^{0.5n}$.

(b) By guess and check, you have to select a value for c and n that makes $1 \leq 2^{cn}$.

When $c = 0.9$ and $n = 3$, $2^{cn} = 2^{2.7}$.
With these values for c and n, $1 \leq 2^{2.7}$.

(c) We can first find a value for c by substituting 2^c for n in the inductive proof from part (a).
Start by replacing n with 2^c:

$$F(2^c) \quad \geq \quad 2^{0.5(2^n)}$$

Problem 4

Is there a faster way to compute the nth Fibonacci number than by fib2 (page 13)? One idea involves matrices.We start by writing the equations and in matrix notation:

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

Similarly,

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} F_0 \\ F_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \cdot \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

and in general,

$$\begin{bmatrix} F_n \\ F_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \cdot \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

(a) So, in order to compute F_n, it suffices to raise this 2×2 matrix, call it X, to the nth power. Show that two 2×2 matrices can be multiplied using 4 additions and 8 multiplications. But how many matrix multiplications does it take to compute X_n?

Take two 2×2 matrices: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$.

When multiplying these two matrices you wind up with the following matrix:

$$\begin{bmatrix} 1(5) + 2(7) & 1(6) + 2(8) \\ 3(5) + 4(7) & 3(6) + 4(8) \end{bmatrix}$$

Based on this operation, you can see that for a $n \times n$ matrix
there are $2n$ additions and $4n$ multiplications.
Essentially, the amount of multiplications is twice the amount of additions
and the amount of additions is twice the amount of elements in a row or column for a $n \times n$ matrix.

(b) Show that $O(\log n)$ matrix multiplications suffice for computing X^n.

Say we have a matrix X we want to multiply 8 times.
We are going to prove that it will take $n\log n$ multiplications to compute X^8.
The multiplications will look like:
$(1st * 2nd) * (3rd * 4th) * (5th * 6th) * (7th * 8th)$
And would result in:
$(X^2) * (X^2) * (X^2) * (X^2)$
By this logic, there are 2^9 multiplications.