Homework 1

RT Hatfield

31 August 2016

1. Problems from book
   (a) $f = \Theta(g)$
   (b) $f = \Omega(g)$
   (c) $f = O(g)$
   (d) $f = \Theta(g)$
   (e) $f = O(g)$
   (f) $f = \Omega(g)$
   (g) $f = O(g)$

2. Bonus
$$g(n) = \frac{1 - c^{n+1}}{1 - c} \tag{1}$$
   (a)
$$\lim_{n\to\infty} \frac{\frac{1-c^{n+1}}{1-c}}{1} = \frac{\frac{1}{1-c}}{1} = \frac{1}{1-c}$$
   If $0 < c < 1$, then $0 < \frac{1}{1-c} < 1$, and $g(n) = \Theta(1)$.

   (b) If $c = 1$, then the sum of the geometric series is $1 + n$.
$$\lim_{n\to\infty} \frac{1+n}{n} = \lim_{n\to\infty} \frac{1}{n} + 1 = 1$$
   Therefore, if $c = 1$, $g(n) = \Theta(n)$.

   (c)
$$\lim_{n\to\infty} \frac{\frac{1-c^{n+1}}{1-c}}{c^n} = \lim_{n\to\infty} \frac{1 - c^{n+1}}{c^n - c^{n+1}} = -\frac{c}{1-c}$$
   If $c > 1$, then $0 < -\frac{c}{1-c} < 1$, and $g(n) = \Theta(c^n)$.

3. Fabonacci Series

(a)

Data: n: desired Fabonacci number in sequence
Result: Fabonacci number at that index
if $n = 0, 1, 2$ **then**
  return 1;
end
else
  return Fabonacci$(n-1)$ + Fabonacci$(n-2)$ $*$ Fabonacci$(n-3)$;
end

This algorithm requires $O(3^n)$. The call stack would look like a ternary tree, which Wikipedia tells me has a size of $\frac{3^{n+1}-1}{2}$ nodes if the height is n. Reducing this to normal Big-O gives $O(3^n)$.

(b)

Data: n: desired Fabonacci number in sequence
Result: Fabonacci number at that index
create array Fab at least as long as n;
for $i \leftarrow 0$ **to** n **do**
  if $i = 0, 1, 2$ **then**
    $Fab[i] \leftarrow 1$;
  end
  else
    $Fab[i] \leftarrow Fab[i-1] + Fab[i-2] * Fab[i-3]$;
  end
end
return $Fab[n]$;

This algorithm will require $2(n-3)$ adds and multiplies to calculate the given Fabonacci number.