University of Houston

COSC 3320: Algorithms and Data Structures
Spring 2016

Solutions for Homework 2

1. (a) Prove that the function $f(n) = na^{\log n}$, where a is a constant greater than 1, is $\Theta(n^c)$ for some constant c.

   (b) Prove that the function $f(n) = n^{1/\log n}$ is $O(1)$.

   (c) Prove that for any constant $a > 0$, $f(n) = \log n$ is $o(n^a)$.

   (d) Order the following functions by order of growth, that is, find an arrangement $f_1, f_2, \ldots, f_{20}$ of the functions such that $f_1 = O(f_2), f_2 = O(f_3), \ldots, f_{19} = O(f_{20})$. (Here $\log n$ means $\log_2 n$.)

n^2	$\dfrac{1}{\log n}$	$n^{4/5}$	1.5^n	$\dfrac{2^{\log n}}{2}$
$n \log\log n$	$\sqrt{\log n}$	$n^{\log_2 3}$	8	$\log\log\log n$
$\sqrt{n^5}$	$\log^{11/6} n$	$e^{\sqrt{n}}$	$\log\log n^3$	$\log n!$
$2^{\sqrt{\log n}}$	$\dfrac{n}{\log n}$	$\log\left(\dfrac{n}{\log n}\right)$	$\dfrac{\log n}{n}$	$n!$

Solution:

(a) Observe that $f(n) = na^{\log n} = n(2^{\log_2 a})^{\log n} = n(2^{\log n})^{\log_2 a} = n^{1+\log 2 \cdot \log_2 a}$. Hence it suffices to set $c = 1 + \log 2 \cdot \log_2 a$, and $c_1 = c_2 = 1$ and $n_0 = 0$ in the definition of $\Theta(\cdot)$

(b) Observe that $f(n) = n^{1/\log n} = (2^{\log_2 n})^{1/\log n} = 2^{\frac{\log_2 n}{\log n}} = 2^{\frac{\log n}{\log 2 \log n}} = 2^{\frac{1}{\log 2}}$. Hence it suffices to set $c = 2^{\frac{1}{\log 2}}$ and $n_0 = 0$ in the definition of $O(\cdot)$.

(c) It's useful to resort to the property seen in class that involves the limit. Since, for any constant $a > 0$, $\lim_{n\to\infty} \frac{\log n}{n^a} = 0$, then $f(n) = \log n$ is $o(n^a)$.

(d) $\frac{\log n}{n}, \frac{1}{\log n}, 8, \log\log\log n, \log\log n^3, \sqrt{\log n}, \log\left(\frac{n}{\log n}\right), \log^{11/6} n, 2^{\sqrt{\log n}}, n^{4/5}, \frac{n}{\log n}, \frac{2^{\log n}}{2}, n\log\log n, \log n!, n^{\log_2 3}, n^2, \sqrt{n^5}, e^{\sqrt{n}}, 1.5^n, n!$

2. Design recursive algorithms for the following problems:

   (a) Compute the n-th Fibonacci number F_n. Recall that the n-th Fibonacci number is defined as follows.

$$F_n = \begin{cases} 1 & \text{if } n = 0, 1, \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2. \end{cases}$$

(b) Compute the n-th power of a number x, x^n, with n non-negative integer. The algorithm should be designed in such a way that it is possible to write a recurrence relation for the total number of multiplications executed by the algorithm for which the Master Theorem applies. Write such a recurrence and apply the Master Theorem to obtain an asymptotic bound for it.

Solution:

(a)

Algorithm 1 Rec_Fib(n)

if $n = 0$ or $n = 1$ **then**
    return 1
return Rec_Fib($n-1$) + Rec_Fib($n-2$)

(b) We shall use the following simple observation:

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & \text{if } n \text{ is even,} \\ x \cdot x^{\lfloor n/2 \rfloor} \cdot x^{\lfloor n/2 \rfloor} & \text{if } n \text{ is odd.} \end{cases}$$

The algorithm is as follows.

Algorithm 2 Power(x, n)

if $n = 0$ **then**
    return 1
if $n = 1$ **then**
    return x
if n mod $2 = 0$ **then**
    return Power($x, n/2$) $\times$ Power($x, n/2$)
else
    return $x \times$ Power($x, \lfloor n/2 \rfloor$) $\times$ Power($x, \lfloor n/2 \rfloor$)

The complexity of Power(x, n) is described by the following recurrence:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 0, 1, \\ 2T(\lfloor n/2 \rfloor) + \Theta(1) & \text{if } n \geq 2. \end{cases}$$

We are in the first case of the Master Theorem, hence the complexity of Power(x, n) is $\Theta(n)$.

3. When possible, apply the Master Theorem to give asymptotic bounds for $T(n)$ for the following recurrences:

(a)

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 3T(n/3) + n/2 & \text{if } n > 1. \end{cases}$$

(b)

$$T(n) = \begin{cases} 4 & \text{if } n = 1, \\ 4T(n/2) + 16n^{15/7} & \text{if } n > 1. \end{cases}$$

(c)

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ T(n/2 + 2) + n^2 & \text{if } n > 1. \end{cases}$$

(d)

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 4T(n/2) + n/\log n & \text{if } n > 1. \end{cases}$$

(e)

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ \log n \cdot T(n/2) + n^2 & \text{if } n > 1. \end{cases}$$

(f)

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n/\log n & \text{if } n > 1. \end{cases}$$

Solution:

(a) $T(n) = \Theta(n \log n)$ (case 2).

(b) $T(n) = \Theta(n^{15/7})$ (case 3).

(c) Master Theorem does not apply (not in the form $T(n/b)$).

(d) $T(n) = \Theta(n^2)$ (case 1).

(e) Master Theorem does not apply (a term not a constant).

(f) Master Theorem does not apply (non-polynomial difference between $n/\log n$ and n).