Noah Hayek (27080409) | Assignment 1 - COMP 352

Wed, May 11, 2016 12:57 AM

A. Written Questions (50 marks):

QA.1

a) Using the relationship among typical growth-rate functions seen in class, order the following functions in non-decreasing order.

$2n\log(n^2), 5n^6, 2^{2013}, 2.5^n, 2.2^n, 2n^{6.5}, \log(n^{10}), 4\cdot\log(n), 2^{100}, n^{1.03}, 70n, n\log n, 8n^6+5n^2$

Ans: 2^{100}, 2^{2013}, $4.\log(n)$, $\log(n^{10})$, $70n$, $n\log n$, $2n\log(n^2)$, $n^{1.03}$, $5n^6$, $8n^6+5n^2$, $2n^{6.5}$, 2.2^n, 2.5^n.

b) Indicate those functions that are "big-Theta" of each other.

Ans: 2^{100} and 2^{2013} are "big-Theta" of each other
$4.\log(n)$ and $\log(n^{10})$ are "big-Theta" of each other
$n\log n$ and $2n\log(n^2)$ are "big-Theta" of each other
$5n^6$ and $8n^6+5n^2$ are "big-Theta" of each other
2.2^n and 2.5^n are "big-Theta" of each other

QA.2

What is the *big-O (O(n))* and *Big-Omega (Ω(n))* time complexity of the *Count(A, B, n)* algorithm *below* in terms of n? Show all necessary steps of how you computed the complexity.

Algorithm *Count(A,B,n)*
Input: Arrays A and B of size n, where n is even.
A, B store integers.
$i \leftarrow 0$ 1 +
$sum \leftarrow 0$ 1 +
while $i < \frac{n}{2}$ **do** $\frac{n}{2}$ +
if $A[i+\frac{n}{2}] < 0$ **then** $\frac{n}{2}\cdot\left(\frac{n}{2}+1\right)$
for $j \leftarrow i + \frac{n}{2}$ **to** n **do**
$sum \leftarrow sum + B[j]$
$i \leftarrow i+1$
return sum 1

* $\frac{n}{2}$ is used everywhere, n is never used

Ans:

$$\frac{\frac{n}{2}\left(\frac{n}{2}+1\right)}{2} = \frac{\frac{n^2}{4}+\frac{n}{2}}{2} = \frac{n^2}{2}+n = \frac{1}{2}n^2+n \Rightarrow O(n^2)\ since\ n^2\ is\ the\ highest$$

$$order\ of\ \frac{1}{2}n^2+n$$

$$therefore, the\ big\ O\ is\ O(n^2)\ and\ the\ big\ Omega\ is\ \Omega(n^2).$$

QA.3

In this question part (a) and (b) are independent.

a) For each of the following pairs of functions, either *f(n)* is in *O(g(n))*, *f(n)* is in *Ω (g(n))*, or *f(n)* is in *θ(g(n))*. For each pair, determine which relationship is correct. Justify your answer.

i) $f(n) = \log n^2$; $g(n) = \log n + 5$.

Ans:

$$f(n) = 2\ log(n) \quad - \quad O(g(n)) = O(log(n))$$

$$f(n) = 2\ log(n)\ is\ O(log(n))\ if\ c > 0\ and\ n_0 \geq 1\ such\ that\ 2\ log(n) \leq c\ log(n)$$
$$for\ n \geq n_0.$$
$$\Rightarrow this\ is\ true\ for\ c = 2\ and\ n_0 = 2.$$

$$f(n) = 2\ log(n)\ is\ \Omega(log(n))\ if\ c > 0\ and\ n_0 \geq 1\ such\ that\ 2\ log(n) \geq c\ log(n)$$
$$for\ n \geq n_0.$$
$$\Rightarrow this\ is\ true\ for\ c = 2\ and\ n_0 = 2.$$

$$\mathbf{f(n) = 2\ log(n)\ is\ \Theta(log(n))\ since\ f(n)\ is\ \Omega(log(n))\ and\ O(log(n)).}$$

ii) $f(n) = \sqrt{n}$; g(n)=log n^2.

Ans:

$$f(n) = n^{\frac{1}{2}} \quad - \quad g(n) = 2\ log(n)$$

$$O(f(n)) = O(n^{\frac{1}{2}}) \quad - \quad O(g(n)) = O(log(n))$$

$$\Omega(f(n)) = \Omega(n^{\frac{1}{2}}) \quad - \quad \Omega(g(n)) = \Omega(log(n))$$

$$O(log(n)) \subset O(n^{\frac{1}{2}}) \ \therefore\ f(n) = n^{\frac{1}{2}}\ is\ in\ O(g(n)) = O(log(n)).$$

$$O(log(n)) \subset O(n^{\frac{1}{2}})\ but\ they\ have\ different\ orders.$$

$$This\ means\ that\ \Omega(log(n)) \not\subset \Omega(n^{\frac{1}{2}})\ since\ the\ order\ of$$
$$the\ elements\ of\ the\ Big\ \Omega\ hierarchy\ is\ the\ reverse\ of\ that\ of\ Big\ O.$$
$$\therefore f(n)\ is\ not\ \Theta(g(n))\ since\ f(n)\ is\ not\ in\ \Omega(g(n)).$$

iii) f(n) = $\log^2$ n; g(n) = log n.

Ans:

$$O(f(n)) = O(log^2 n) \quad - \quad O(g(n)) = O(log(n))$$
$$\Omega(f(n)) = \Omega(log^2 n) \quad - \quad \Omega(g(n)) = \Omega(log(n))$$
$$To\ determine\ the\ order\ of\ log^2 n\ let\ m = log(n),$$
$$this\ means\ that\ m^2 = (log(n))^2 = log^2 n\ and\ is\ of\ higher\ order\ than\ m = log(n).$$
$$\therefore\ O(log(n)) = O(g(n)) \subset O(f(n)) = O(log^2 n)\ and\ f(n)\ is\ in\ O(g(n)).$$
$$The\ order\ of\ the\ elements\ of\ the\ big\ \Omega\ hierarchy\ is\ the\ reverse\ of\ that\ of\ big\ O.$$
$$This\ means\ that\ \Omega(f(n)) = \Omega(log(n)) \not\subset \Omega(log^2 n) = \Omega(g(n)).$$
$$\therefore f(n)\ is\ not\ \Theta(g(n))\ since\ f(n)\ is\ not\ in\ \Omega(g(n)).$$

iv) f(n) = n; g(n) = $\log^2$n.

Ans:

$$O(f(n)) = O(n) \quad - \quad O(g(n)) = O(log^2 n)$$
$$let\ n = 700000\ (large\ value) \Rightarrow log^2(700000) \approx 19.41.$$
$$Clearly\ g(n) = log^2 n\ is\ more\ efficient\ then\ f(n) = n.$$
$$\therefore O(log^2 n) \subset O(n)\ and\ f(n)\ is\ in\ O(g(n)).$$
$$f(n)\ is\ not\ in\ \Omega(g(n))\ because\ the\ order\ of\ the\ elements\ of$$
$$the\ \Omega\ hierarchy\ is\ the\ reverse\ of\ that\ of\ Big\ O.$$
$$\therefore f(n)\ is\ not\ in\ \Omega(g(n))\ and\ not\ in\ \Theta(g(n)).$$

v) f(n) = n log n + n; g(n) = log n.

Ans:

$$O(f(n)) = O(n\ log(n)) \quad - \quad O(g(n)) = O(log(n))$$
$$f(n)\ is\ in\ O(g(n))\ since\ O(log(n)) \subset O(n\ log(n)).$$
$$On\ the\ other\ hand,\ f(n)\ is\ not\ in\ \Omega(g(n))\ since\ Omega(log(n)) \not\subset \Omega(n\ log(n))$$
$$as\ the\ order\ of\ the\ element\ of\ the\ \Omega\ hierachy\ is\ the\ reverse\ of\ that\ of\ Big\ O$$
$$and\ based\ on\ that,\ we\ can\ conclude\ that\ f(n)\ is\ not\ in\ \theta(g(n)).$$

vi) f(n) = log n^2; g(n) = $(\log n)^2$.

Ans:

$$f(n) = 2\ log(n)$$
$$O(f(n)) = O(log(n)) \quad - \quad O(g(n)) = O(log^2).$$
$$let\ x = log(n) \Rightarrow x^2 = log^2 n;\ O(log^2 n) = O(x^2) \not\subset O(x) = (log(n)).$$
$$\therefore f(n)\ is\ not\ in\ O(g(n))\ and\ f(n)\ is\ not\ in\ \Theta(g(n)).$$
$$However,\ \Omega(log^2) = \Omega(x^2) \subset \Omega(log(n)) = \Omega(x)\ and\ this\ indicates\ that\ f(n)\ is\ in\ \Omega(g(n)).$$

vii) f(n) = 10; g(n) = log 10.

Ans:

$$O(f(n)) = O(1) \quad - \quad O(g(n)) = O(1),$$
$$since\ both\ f(n)\ and\ g(n)\ are\ constant\ functions,\ they\ have\ O(1)\ and\ \Omega(1)\ and\ \Theta(1).$$
$$\therefore f(n)\ is\ in\ \Theta(g(n))\ and\ this\ indacates\ that\ f(n)\ is\ also\ in\ both\ O(g(n))\ and\ \Omega(g(n)).$$

Noah Hayek (27080409) | Assignment 1 - COMP 352

Sun, May 15, 2016 12:48 AM

viii) $f(n) = 2^n$; $g(n) = 10n^2$.

Ans:

$$O(f(n)) = O(2^n)\ and\ O(g(n)) = O(n^2) \Rightarrow \Omega(f(n)) = \Omega(2^n)\ and\ \Omega(g(n)) = \Omega(n^2).$$
$$f(n)\ is\ in\ O(g(n))\ since\ O(n^2) \subset O(2^n).\ But\ f(n)\ is\ not\ in\ \Omega(g(n))\ since \Omega(n^2) \not\subset \Omega(2^n)$$
$$and\ based\ on\ that,\ f(n)\ is\ not\ in\ \Theta(g(n)).$$

ix) $f(n) = 2^n$; $g(n) = n \log n$.

Ans:

$$O(f(n)) = O(n^2)\ and\ O(g(n)) = O(n\ log(n)) \Rightarrow \Omega(f(n)) = \Omega(n^2)\ and\ \Omega(g(n)) = \Omega(n\ log(n))$$
$$f(n)\ is\ in\ O(g(n))\ since\ O(n\ log(n)) \subset O(n^2).$$
$$But\ f(n)\ is\ not\ in\ \Omega(g(n))\ since \Omega(n\ log(n)) \not\subset \Omega(n^2)$$
$$and\ based\ on\ that,\ f(n)\ is\ not\ in\ \Theta(g(n)).$$

x) $f(n) = 2^n$; $g(n) = 3^n$.

Ans:

$$O(f(n)) = O(2^n)\ and\ O(g(n)) = O(3^n) \Rightarrow \Omega(f(n)) = \Omega(2^n)\ and\ \Omega(g(n)) = \Omega(3^n).$$
$$2^n\ and\ 3^n\ are\ two\ exponential\ functions\ with\ different\ base\ numbers\ and\ therefore,$$
$$are\ not\ of\ the\ same\ order\ since\ 2^n < 3^n\ for\ n > 0.$$
$$Based\ on\ that\ O(3^n) \not\subset O(2^n)\ and\ f(n)\ is\ not\ in\ O(g(n))\ and\ is\ not\ in\ \Theta(g(n)).$$
$$On\ the\ other\ hand,\ \Omega(3^n) \subset \Omega(2^n)\ since\ the\ \Omega\ hierarchy\ of\ elements\ is\ the\ reverse$$
$$of\ that\ of\ Big\ O.$$
$$\therefore f(n)\ is\ in\ \Omega(g(n)).$$

xi) $f(n) = 2^n$; $g(n) = n^n$.

Ans:

$$O(f(n)) = O(2^n)\ and\ O(g(n)) = O(n^n) \Rightarrow \Omega(f(n)) = \Omega(2^n)\ and\ \Omega(g(n)) = \Omega(n^n).$$
$$f(n)\ is\ not\ in\ O(g(n))\ because\ n^n\ is\ of\ higher\ order\ than\ 2^n.$$
$$This\ also\ indicates\ that\ f(n)\ is\ not\ in\ \Theta(g(n)).$$
$$However,\ since\ the\ hierarchy\ of\ elements\ of\ \Omega\ is\ the\ inverse\ of\ that\ of\ Big\ O,$$
$$\Omega(2^n)\ is\ in\ \Omega(n^n)\ since\ \Omega\ (n^n) \subset \Omega(2^n).$$

b) The following run times were obtained when using two different algorithms on a data set of size *n*. Based on this timing data, guess at the asymptotic complexity of each algorithm as a function of *n*. Use Big-O notation in its simplest form and briefly explain how you reached your conclusion.

i)

n	Execution Time (in seconds)
1000	0.743
2000	3.021
4000	12.184
8000	50.320

Ans: O(n), since the execution time increases by a factor of approximately 4 when the value of n is doubled.

ii)

n	Execution Time (in microseconds or millionths of a second)
1000	0.01
1000000	20
1000000000	30000

Ans: O(log(n)), since the execution prove that the algorithm is effective.