HW2

Problem 1 From Victor M Espaillat

1. **O, Ω, Θ Notation Practice**: (15 points)
Provide either a proof (using definition) to support the claim or a counter example to disprove it.

(1) $2^{n+1} = O(2^n)$
Can 2^n provide an upper bound for 2^{n+1}?
Does there exist positive constants c and n_0 such that $(0 \leq 2^{n+1} \leq c \cdot 2^n)\ \forall\, n \geq n_0$?

$2^{n+1} \leq c \cdot 2^n$
$2 \cdot 2^n \leq c \cdot 2^n$
$c \geq 2$

True.
$c \cdot 2^n$ will provide an upper bound for $2^{n+1}\ \forall\, c \geq 2$.

(2) $f(n) = \Theta(f(n/2))$
Can $f(n/2)$ provide a tight bound for $f(n)$ where f is any function.

Proof by counterexample:
Let $f(n) = 2^n$
$2^n = \Theta(2^{n/2})$?

Big-O case:
$2^n = O(2^{n/2})$?
Can $2^{n/2}$ provide an upper bound for 2^n?
Does there exist positive constants c and n_0 such that $(0 \leq 2^n \leq c \cdot 2^{n/2})\ \forall\, n \geq n_0$?

$2^n \leq c \cdot 2^{n/2}$
$\frac{2^n}{2^{n/2}} \leq c$
$2^{n/2} \leq c$

False.
This is impossible to solve. We cannot find a constant c such that $c \geq 2^{n/2}$.

(3) $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$

True.

If $f(n) = O(g(n))$ then ∃ positive constants c_1, n_0 such that $\forall\, n \geq n_0$ the following inequality holds:

$0 \leq f(n) \leq c_1 \cdot g(n)$

$0 \leq \frac{1}{c_1} f(n) \leq g(n)$

$0 \leq k \cdot f(n) \leq g(n)$ Let $k = \frac{1}{c_1}$

If $g(n) = \Omega(f(n))$ then ∃ positive constants c_2, n_0 such that $\forall\, n \geq n_0$ the following inequality holds:

$0 \leq c_2 \cdot f(n) \leq g(n)$

$0 \leq k \cdot f(n) \leq g(n)$ k is a positive constant, so we can let $c_2 = k$

$f(n) = O(g(n))$ being true implies that $g(n) = \Omega(f(n))$ is also true because the inequality that needs to hold true for $O(g(n))$ is the same one that needs to hold true for $\Omega(f(n))$.

2. Victor M Espaillat

2. **Function Order of Growth**: (20 points)
List the 4 functions below in nondecreasing asymptotic order of growth.
Justify your answer mathematically by showing values of c and n_0 for each pair of functions that are adjacent in your ordering.

$$(\lg n)^2 \qquad n^{-2} \qquad 2n^2 \qquad \lg\left(2^{\lg(n^2)}\right)$$

$$O(g(n)) = \begin{cases} f(n) : \exists \text{ constants } c > 0, n_0 > 0 \\ \quad \text{s.t. } 0 \le f(n) \le c \cdot g(n) \; \forall \, n \ge n_0 \end{cases}$$

Claim: $n^{-2} \le \lg\left(2^{\lg(n^2)}\right)$

$n^{-2} \le \lg(n^2)$

$n^{-2} \in O(\lg n^2) \Rightarrow 0 \le n^{-2} \le c \cdot \lg n^2$

$0 \le \frac{1}{n^2} \le \lg n^2$ Let $c = 1$

$0 \le 1 \le n^2 \lg n^2$

$n_0 = 2$ because the inequality holds $\forall \, n \ge 2$

Claim: $\lg n^2 \in O((\lg n)^2) \Rightarrow 0 \le \lg n^2 \le c \cdot (\lg n)^2$

$0 \le \lg n^2 \le (\lg n)^2$ Let $c = 1$

$n_0 = 4$

Claim: $(\lg n)^2 \in O(2n^2) \Rightarrow 0 \le (\lg n)^2 \le c \cdot 2n^2$

$0 \le (\lg n)^2 \le 4n^2$ Let $c = 2$

$0 \le \lg n \le 2n$

$n_0 = 1$

Conclusion:

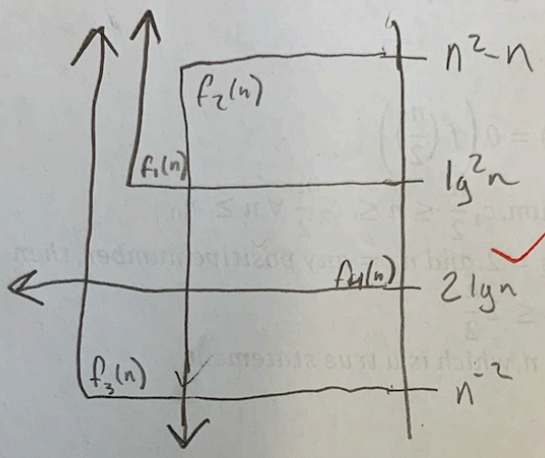
Smallest Largest

$$n^{-2} \le \lg\left(2^{\lg(n^2)}\right) \le (\lg n)^2 \le 2n^2$$

3 from Tom Clunie

3)

a)



b) TRUE. Because $f_4(n) \in \theta\left(\lg\left(2^{\lg(n^2)}\right)\right)$, *and* $\lg\left(2^{\lg(n^2)}\right)$ *can be bounded by* $O((lgn)^2)$ $\lg\left(2^{\lg(n^2)}\right) \leq (lgn)^2$ *for any* $n \geq 1$ *(as shown in #2)and* $(lgn)^2$ *is the lower bound of* f_1 *, therefore* $f_4(n)$*will be able to be bounded by any* $O(f_1(n))$

c) FALSE. We have no information on the lower bound of f_2. Suppose $f_2(n)$ is defined as $\frac{1}{n^3}$. In this case, $f_2(n) \in O(n^2 - n)$ but $f_2(n) \notin \Omega(f_3(n))$, as $\frac{1}{n^3} \leq \frac{1}{n^2}\ \forall\ n \geq 1$.

Example: $n = 2, f_2(2) = \frac{1}{8}, f_3(2) = \frac{1}{4}$. $f_2(2) < f_3(2)$, $f_2(n) = \frac{1}{n^3}$, $f_3(n) = \frac{1}{n^2}$

d) FALSE. We have no information on the upper bound of f_1. Suppose $f_1(n)$ is defined as n^2. In this case, $f_1(n) \in \Omega((lgn)^2)$ but $f_1(n) \notin O(f_2(n))$, as $n^2 \geq n^2 - n\ \forall\ n \geq 0$

Example: $n = 1, f_1(1) = 1, f_2(1) = 0$. $f_1(1) > f_2(1)$, $f_1(n) = n^2$, $f_2(n) = n^2 - n$

e) FALSE. First we compare between $\lg^3 n$ *and* $\lg\left(2^{\lg(n^2)}\right) = \lg(n^2)$, and find that $\lg^3 n > \lg(n^2)\ \forall\ n > \sim 4$. Because no c *and* n_0 can be found where $\lg(n^2) \geq \lg^3 n$, we can state that $f_4(n) \notin \theta(\lg^3 n)$

Example: $n = 4, f_4(4) = 4, \lg^3(4) = 8$. As n increases, the difference grows exponentially, so no c nor n_0 can be found to satisfy the definition

Problem 4&5 From Victor M Espaillat

4. **Analysis:** (10 points)
Your client is developing two new algorithms. $f_1(n)$ and $f_2(n)$ are the worst-case running time for these two algorithms: $f_1(n) = n \lg n$, and $f_2(n) = 512n$. As a consultant, which algorithm will you recommend to your client? Justify your answer. (Hint: Please consider the asymptotical growth of the functions and also consider the reality.)

$512n$ is asymptotically smaller than $n \lg n$; it grows slower with n, therefore it is faster.

$$\lim_{n\to\infty} \frac{512n}{n \lg n} = \lim_{n\to\infty} \frac{512}{\lg n} = 0$$

However, $512n$ is only faster than $n \lg n$ after a certain, extremely large, value of n.

$$\begin{aligned} 512n \in O(n \lg n) \Rightarrow 0 \leq 512n &\leq c \cdot n \lg n \\ 0 \leq 512n &\leq n \lg n \qquad \text{Let } c = 1 \\ 0 \leq 512 &\leq \lg n \\ 0 \leq 2^{512} &\leq n \\ n_0 &= 2^{512} \end{aligned}$$

Since it is unlikely that the client will work with such large values of $n \geq 2^{512}$ I would recommend the $n \lg n$ algorithm.

5. **Pseudocode Analysis:** (25 points)
For the pseudocode below for procedure Mystery(n), derive tight upper and lower bounds on its asymptotic *worst-case* running time $f(n)$. That is, for the set of inputs including those that force Mystery to work its hardest, find $g(n)$ such that $f(n) \in \Theta(g(n))$. Assume that the input n is a positive integer. Justitfy your answer.

Mystery(n)	Cost	Times
1. if n is an even number	c_1	1
2. for i = 1 to n	c_2	$n+1$
3. for j = n/2 to n	c_3	$n(n/2+1)$
4. print "even"	c_4	$n(n/2)$
5. else	c_5	0
6. for k = 1 to n/4	c_6	$(n/4)+1$
7. for m = 1 to n	c_7	$(n/4)(n+1)$
8. print "odd"	c_8	$(n/4)n$

Worst-case:

$$T(n) = c_1 + c_2(n+1) + c_3 n\left(\frac{n}{2}+1\right) + c_4 n\left(\frac{n}{2}\right)$$

$$= c_1 + c_2 n + c_2 + \frac{c_3 n^2}{2} + c_3 n + \frac{c_4 n^2}{2}$$

$$= \left(\frac{c_3}{2} + \frac{c_4}{2}\right) n^2 + (c_2 + c_3)n + (c_1 + c_2)$$

$$= an^2 + bn + c$$

$$= O(n^2)$$

Best-case:

$$T(n) = c_6\left(\frac{n}{4}+1\right) + c_7\left(\frac{n}{4}\right)(n+1) + c_8\left(\frac{n}{4}\right)n$$

$$= \frac{c_6 n}{4} + c_6 + \frac{c_7 n^2}{4} + \frac{c_7 n}{4} + \frac{c_8 n^2}{4}$$

$$= \left(\frac{c_7}{4} + \frac{c_8}{4}\right) n^2 + \left(\frac{c_6}{4} + \frac{c_7}{4}\right) n + c_6$$

$$= an^2 + bn + c$$

$$= \Omega(n^2)$$

$$\boxed{T(n) = \Theta(n^2)}$$

100

*not really worst-case / best-case, the first one is when n is an even number; the second calcuation is whne n is an odd number. No matter which case, the worst-case running time is $\theta(n^2)$