CS521 Fall 2011 \ Assignment #1

Ariel Stolerman

Page 61, problem 3-3:

***a*.**

The functions g_i are ranked by order of growth from highest to lowest (top-bottom, left to right). Functions in the same equivalence class ($g_i = \Theta(g_j)$ and vice-versa) are placed in the same row:

$2^{2^{n+1}}$	$(\lg n)^{\lg n}, n^{\lg \lg n}$	$\lg^2 n$
2^{2^n}	$(\lg n)!$	$\ln n$
$(n+1)!$	n^3	$\sqrt{\lg n}$
$n!$	$n^2, 4^{\lg n}$	$\ln \ln n$
e^n	$n \lg n, \lg(n!)$	$2^{\lg^* n}$
$n \cdot 2^n$	$n, 2^{\lg n}$	$\lg^* \lg n, \lg^* n$
2^n	$(\sqrt{2})^{\lg n}$	$\lg \lg^* n$
$\left(\frac{3}{2}\right)^n$	$2^{\sqrt{2 \lg n}}$	$1, n^{\frac{1}{\lg n}}$

***b*.**

An example for a non-negative function f such that $f \notin \{O(g_i)\} \cup \{\Omega(g_i)\}$ for all g_i from section ***a***:

$$f(n) = 2^{2^{2^n}} \cdot (\sin \frac{\pi n}{2} + 1)$$

Explanation: $\sin \frac{\pi n}{2} \in [-1,1] \Rightarrow \left(\sin \frac{\pi n}{2} + 1\right) \in [0,2]$. The term $2^{2^{2^n}}$ is non-negative, so the multiplication of the two is non-negative as required. Surely when (i) $\sin \frac{\pi n}{2} + 1 = 2$ we satisfy $f \neq O(g_i)$ for all i, and when (ii) $\sin \frac{\pi n}{2} + 1 = 0$ we satisfy $f \neq \Omega(g_i)$ for all i. Since sine is periodic, asymptotically we have infinite n's for which (i) / (ii) occur. Thus the requirement is satisfied.

Page 62, problem 3-5:

***a*.**

We will define $\overset{\infty}{\Omega}$ as follows:

$f(n) = \overset{\infty}{\Omega}(g(n)) \Leftrightarrow \exists\ positive\ constant\ c\ and\ set\ \mathbb{N}^*\ s.t.\ |\mathbb{N}^*| = \aleph_0\ and\ \forall n \in \mathbb{N}^*: 0 \leq c \cdot g(n) \leq f(n)$

Whereas the definition of Ω is:

$f(n) = \Omega(g(n)) \Leftrightarrow \exists\ positive\ constants\ c, n_0\ s.t. \forall n \geq n_0: 0 \leq c \cdot g(n) \leq f(n)$

i. Proof that either $f(n) = O(g(n))$ or $f(n) = \overset{\infty}{\Omega}(g(n))$:

Suppose that there exist non-negative functions f, g such that $f \notin O(g)$ and $f \notin \overset{\infty}{\Omega}(g)$, then:

1. $\forall$ positive constants c, n_0 $\exists n \geq n_0$: $f(n) > c \cdot g(n)$
2. $\forall$ positive constant c there is NO infinite set of integers $\mathbb{N}^*$ that satisfies $\forall n \in \mathbb{N}^*$: $f(n) \geq c \cdot g(n)$ – i.e. for every positive constant c the number of integers that satisfy $f(n) \geq c \cdot g(n)$ is **finite**.

Let c be some positive constant. According to (2) there is a finite set of integers S such that $\forall n \in S$: $f(n) \geq c \cdot g(n)$. Let $n_0 = \max S + 1$, therefore $\forall n \geq n_0$: $f(n) < c \cdot g(n)$.

But according to (1) for **every** c, n_0 there exists $n \geq n_0$ such that $f(n) > c \cdot g(n)$, contradiction.

Therefore either $f(n) = O(g(n))$ or $f(n) = \overset{\infty}{\Omega}(g(n))$.

ii. Proof that the above does not apply if we substitute $\overset{\infty}{\Omega}$ for Ω:

Choosing $f(n) = 2^n \left(\sin\frac{\pi n}{2} + 1\right), g(n) = n$ immediately proofs that, since:

- $f(n) \neq O(g(n))$: We will satisfy the negation of the condition of the big-O definition:
  $\forall c \forall n_0 \exists n > n_0 \; s.t. f(n) > c \cdot g(n)$ – for every constant c we will choose, and for every breaking point n_0, we can simply take the first $n_1 > n_0$ that satisfies:
  - $\sin\frac{\pi n_1}{2} + 1 \geq 1$
  - $2^{n_1} > c \cdot n_1$

  To achieve $f(n_1) > c \cdot g(n_1)$, thus concluding that $f(n) \neq O(g(n))$.
- $f(n) \neq \Omega(g(n))$: in the same matter we will satisfy the negation of the condition of the big-Ω definition:
  $\forall c \forall n_0 \exists n > n_0 \; s.t. f(n) < c \cdot g(n)$ – for every constant c chosen, and for every break point n_0, we can choose the first $n_1 > n_0$ that satisfies $\sin\frac{\pi n_1}{2} + 1 = 0$ (and $n_1 > 0$) to achieve $f(n_1) < c \cdot g(n_1)$, thus concluding that $f(n) \neq \Omega(g(n))$.

But $f = \overset{\infty}{\Omega}(g)$ since for every constant c there is an infinite set $\mathbb{N}^*$ that satisfy: $\forall n \in \mathbb{N}^*$: $0 \leq c \cdot g(n) \leq f(n)$ – simply set $\mathbb{N}^* = \left\{n \mid n > n_0 \; and \sin\frac{\pi n}{2} > 0, where \; n_0 \; satisfies \; \forall n_1 > n_0 : 2^{n_1} > c \cdot n_1\right\}$, i.e. an infinite set of positive integers that assure $f(n) \geq c \cdot g(n)$.

Therefore we have proven that the rule in (i) does not apply when substituting $\overset{\infty}{\Omega}$ for Ω.

b.

First we notice that every non-negative function g satisfies $\Omega(g) \subseteq \overset{\infty}{\Omega}(g)$, which means that the $\overset{\infty}{\Omega}$ notation is "weaker" than Ω (in the sense of "less strict"), thus giving us less information about the running time of function in that set, which is a disadvantage. More specifically, the notation $f = \overset{\infty}{\Omega}(g)$ accounts for specific characteristics of f only for certain areas in the domain of f.

On the other hand, $f = \overset{\infty}{\Omega}(g)$ means that f is bounded below by g at least intermittently, and sometimes this knowledge may be the only knowledge that can be derived from the data, and perhaps knowing a lower bound for f in parts of its domain is sufficient for our requirements (better than not knowing any bound at all – to help decide whether a function fits our needs or not).

c.

Important note: the answer below omits 0 as lower bound for the Θ, O and Ω notations, since if we take them into consideration we get $f(n) = O'\big(g(n)\big) \Rightarrow |f(n)| = O\big(g(n)\big) \Rightarrow 0 \leq f(n) \leq c_1 g(n)$ and $0 \leq -f(n) \leq c_2 g(n) \Rightarrow$ $f(n) \in [-c_2 g(n), 0]$ and $f(n) \in [0, c_1 g(n)] \Rightarrow f(n) \equiv 0$. And this would make the question asked here rather irrelevant…

$f(n) = O'\big(g(n)\big) \Leftrightarrow |f(n)| = O\big(g(n)\big) \Rightarrow$
$f(n) = O\big(g(n)\big)\ and\ \big(-f(n)\big) = O\big(g(n)\big) \Rightarrow$

- $\exists\ positive\ constants\ c_1, n_1\ s.t. \forall n \geq n_1 \colon f(n) \leq c_1 \cdot g(n)$
- $\exists\ positive\ constants\ c_2, n_2\ s.t. \forall n \geq n_2 \colon -f(n) \leq c_2 \cdot g(n) \Rightarrow f(n) \geq -c_2 \cdot g(n)$

Theorem 3.1 is that for every two functions f, g: $\boxed{f(n) = \Theta\big(g(n)\big) \Leftrightarrow f(n) = O\big(g(n)\big)\ and\ f(n) = \Omega\big(g(n)\big)}$.

If we exchange the O notation in theorem 3.1 with the O' notation as defined above, the following happens to the $\Leftrightarrow$ (with respect to the directions above):

$\underline{\Leftarrow}$:

From the two conditions extracted above for O' it can immediately be seen that $f(n) = O'\big(g(n)\big) \Rightarrow f(n) = O\big(g(n)\big)$ (by simply taking the first condition above), and given also $f(n) = \Omega\big(g(n)\big)$ by theorem 3.1 we get that $f(n) = \Theta\big(g(n)\big)$.

Therefore the $\Leftarrow$ direction still holds.

$\underline{\Rightarrow}$:

This direction however is not true, since $f(n) = \Theta\big(g(n)\big)$ doesn't necessarily implies $f(n) = O'\big(g(n)\big)$.

Since the statement is supposed to hold for any f, g we can show a counter example where $f(n) = \Theta\big(g(n)\big)$ but $f(n) \neq O'\big(g(n)\big)$. If we take $f(n) = g(n) = -\frac{1}{n}$:

- Of course $-\frac{1}{n} = \Theta\left(-\frac{1}{n}\right)$
- $\left|-\frac{1}{n}\right| = \frac{1}{n} \neq O'\left(-\frac{1}{n}\right)$ because there exists no such **positive** c that holds: $\frac{1}{n} \leq c\left(-\frac{1}{n}\right)$

Therefore the $\Rightarrow$ direction does not hold.

d.

Definition of $\tilde{O}$: $f(n) = \tilde{O}\big(g(n)\big) \Leftrightarrow \exists\ positive\ constants\ c, k, n_0\ s.t. \forall n \geq n_0 \colon 0 \leq f(n) \leq c \cdot g(n) \cdot \lg^k(n)$

Corresponding definition for $\tilde{\Omega}$ and $\tilde{\Theta}$:

- $f(n) = \tilde{\Omega}\big(g(n)\big) \Leftrightarrow \exists\ positive\ constants\ c, k, n_0\ s.t. \forall n \geq n_0 \colon 0 \leq c \cdot g(n) \cdot \lg^k(n) \leq f(n)$

- $f(n) = \tilde{\Theta}\big(g(n)\big) \Leftrightarrow \exists\ positive\ constants\ c_1, c_2, k_1, k_2, n_0\ s.t. \forall n \geq n_0$:
  $0 \leq c_1 \cdot g(n) \cdot \lg^{k_1}(n) \leq f(n) \leq c_2 \cdot g(n) \cdot \lg^{k_2}(n)$

Proof for Theorem $\tilde{3.1}$:

$$f(n) = \tilde{\Theta}\big(g(n)\big) \Leftrightarrow f(n) = \tilde{\Omega}\big(g(n)\big)\ and\ f(n) = \tilde{O}\big(g(n)\big)$$

⇒:

Trivial:

If $f(n) = \tilde{\Theta}\big(g(n)\big)$ then there exist positive constants c_1, k_1, n_0 s.t. for all $n \geq n_0$:

$0 \leq c_1 \cdot g(n) \cdot \lg^{k_1}(n) \leq f(n)$, thus by definition $f(n) = \tilde{\Omega}\big(g(n)\big)$.

In the same matter, there also exist positive constants c_2, k_2 (with the same n_0) s.t. for all $n \geq n_0$:

$0 \leq f(n) \leq c_2 \cdot g(n) \cdot \lg^{k_2}(n)$, thus by definition $f(n) = \tilde{O}\big(g(n)\big)$.

⇐:

If both $f(n) = \tilde{\Omega}(g(n)$ and $f(n) = \tilde{O}\big(g(n)\big)$ then there exist positive constants $c_1, k_1, n_1, c_2, k_2, n_2$ s.t.

$$\forall n \geq n_1 \colon 0 \leq c_1 \cdot g(n) \cdot \lg^{k_1}(n) \leq f(n), \qquad \forall n \geq n_2 \colon 0 \leq f(n) \leq c_2 \cdot g(n) \cdot \lg^{k_2}(n)$$

Thus by picking $n_0 = \max\{n_1, n_2\}$ we satisfy:

$$\forall n \geq n_0 \colon 0 \leq c_1 \cdot g(n) \cdot \lg^{k_1}(n) \leq f(n) \leq c_2 \cdot g(n) \cdot \lg^{k_2}(n)$$

And so by definition $f(n) = \tilde{\Theta}\big(g(n)\big)$.

Page 63, problem 3-6:

	$\boldsymbol{f(n)}$	$\boldsymbol{c}$	$\boldsymbol{f_c^*(n)}$
a	$n-1$	0	$\boxed{n}$
b	$\lg n$	1	This is simply the $\boxed{\lg^* n}$ function (which after many tries could not be simplified to a non-recursive definition. That's probably why it has its own notation).
c	$n/2$	1	$\boxed{\lceil \lg n \rceil}$
d	$n/2$	2	$\boxed{\lceil \lg n \rceil - 1}$
e	$\sqrt{n}$	2	$\boxed{\lceil \lg \lg n \rceil}$ since $(\ldots((n^{0.5})^{0.5})^{0.5}\ldots)^{0.5} \leq 2 \Rightarrow n \leq \underbrace{(((2^2)^2)^2\ldots)^2}_{k\ times} = 2^{2^k} \Rightarrow k \geq \lg \lg n$
f	$\sqrt{n}$	1	$\boxed{\begin{cases} 0, & n \leq 1 \\ \infty, & n > 1 \end{cases}}$ -- maybe defining the function in $(-\infty, 1]$ would be more correct
g	$n^{1/3}$	2	$\boxed{\lceil \lg_3 \lg_2 n \rceil}$ since $\Big(\ldots\Big(\underbrace{\big(n^{1/3}\big)^{1/3}}_{k\ times}\Big)^{1/3}\ldots\Big)^{1/3} \leq 2 \Rightarrow n \leq \underbrace{(((2^3)^3)^3\ldots)^3}_{k\ times} = 2^{3^k} \Rightarrow \lg_2 n \leq 3^k \Rightarrow \lg_3 \lg_2 n \leq k$
h	$n/\lg n$	2	After many tries I couldn't solve this one.

Page 108, problem 4-3:

a. $T(n) = 4T\left(\frac{n}{3}\right) + n\lg n$

$a = 4, b = 3, f(n) = n\lg n, \quad 1.261 < \log_3 4 < 1.262 \Rightarrow$

$f(n) = n\lg n = O(n^{1.261}) \underset{\epsilon \cong 0.0008}{=} O\left(n^{\log_3 4 - \epsilon}\right) = O\left(n^{\log_b a - \epsilon}\right) \Rightarrow$ Case 1 of MT applies$\Rightarrow$

$\boxed{T(n) = \Theta\left(n^{\log_3 4}\right)}$ [that is $\cong \Theta(n^{1.2618})$]

b. $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg n}$

Since $a = b = 3$ then $n^{\log_3 3} = n$, which means MT cannot be applied (as there's neither equal order of growth between $n^{\log_b a}$ and $f(n)$ nor one polynomially larger than the other).

Looking at the recursion tree we find:

- At level 0: $f(n) = \frac{n}{\lg n}$
- At level 1: $3f\left(\frac{n}{3}\right) = \frac{3n}{3} / \lg\frac{n}{3} = \frac{n}{\lg n - \lg 3} = \frac{n}{(\lg_3 n - \lg_3 3)/\lg_2 3} = \frac{\lg_2 3 \cdot n}{\lg_3 n - 1}$
- At level 2: $9f\left(\frac{n}{9}\right) = \frac{9n}{9} / \lg\frac{n}{3^2} = \frac{n}{\lg n - 2\lg 3} = \cdots = \frac{\lg_2 3 \cdot n}{\lg_3 n - 2}$

… Until level $\lg_3 n - 1$.

$\Rightarrow T(n) = \sum_{i=0}^{\lg_3 n - 1} \frac{\lg_3 2 \cdot n}{\lg_3 n - i} = n \cdot \lg_3 2 \sum_{i=0}^{\lg_3 n - 1} \frac{1}{\lg_3 n - i} \overset{(*)}{=} n \cdot \underbrace{\lg_3 2}_{constant} \sum_{i=1}^{\lg_3 n} \frac{1}{i} \overset{(**)}{=} \boxed{\Theta(n \lg\lg n)}$

$(*)$: Switching order of summation and general element in the sum accordingly:

$j = \log_3 n - i \Rightarrow \begin{cases} i = 0 \Rightarrow j = \log_3 n \\ i = \log_3 n \Rightarrow j = \log_3 n - (\log_3 n - 1) = 1 \end{cases} \Rightarrow$ the sum after index change is $\log_3 n$ down to 1, or simply 1 up to $\log_3 n$ of the general element $\frac{1}{j}$.

$(**)$: Harmonic series order of growth is logarithmic, i.e. $\sum_{k=1}^{n} \frac{1}{k} = \Theta(\lg n)$, hence $\sum_{k=1}^{\lg n} \frac{1}{k} = \Theta(\lg\lg n)$

c. $T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n}$

$a = 4, b = 2, f(n) = n^2\sqrt{n} = n^{2.5}$

$\begin{cases} f(n) = n^{2+0.5} = \Omega(n^2) = \Omega\left(n^{\log_2 4}\right) \Rightarrow f(n) = \Omega\left(n^{\log_b a + \epsilon}\right) \; for \; \epsilon = 0.5 > 0 \\ a \cdot f\left(\frac{n}{b}\right) = 4f\left(\frac{n}{2}\right) = 4\left(\frac{n}{2}\right)^2\sqrt{\frac{n}{2}} = 4 \cdot \frac{n^2}{4} \cdot \frac{\sqrt{n}}{\sqrt{2}} = \frac{1}{\sqrt{2}} n^2\sqrt{n} = \frac{1}{\sqrt{2}} f(n) \Rightarrow a \cdot f\left(\frac{n}{b}\right) \le c \cdot f(n) \; for \; c = \frac{1}{\sqrt{2}} < 1 \end{cases} \Rightarrow$

Case 3 of MT applies$\Rightarrow$ $\boxed{T(n) = \Theta(n^{2.5})}$

d. $T(n) = 3T\left(\frac{n}{3} - 2\right) + \frac{n}{2}$

The intuition is $\Theta(n \lg n)$ because of the form of the equation, so we'll prove an upper and lower bound.

Upper bound:

$T(n) = 3T\left(\frac{n}{3} - 2\right) + \frac{n}{2} \overset{(*)}{\leq} 3T\left(\frac{n}{3}\right) + \frac{n}{2} \underset{MT\ case\ \#2}{=} \Theta(n \lg n) \Rightarrow T(n) = O(n \lg n)$

$(*)$: The recursion tree on the right to the $\leq$ will have more nodes than the one on the left to the $\leq$.

Lower bound:

We assume correctness for all integers up to n (excluding) to prove the lower bound. Let c denote the constant for that domain:

$T(n) = 3T\left(\frac{n}{3} - 2\right) + \frac{n}{2} \geq 3 \cdot c\left(\frac{n}{3} - 2\right) \lg\left(\frac{n}{3} - 2\right) + \frac{n}{2} = \left(cn - \frac{3}{2}c\right)(\lg(n-6) - \lg 3) + \frac{n}{2} =$

$cn \lg(n-6) - c \lg 3 \cdot n - \frac{3}{2}c \lg(n-6) + \frac{3}{2}\lg 3\, c + \frac{n}{2} \geq$ / $\lg(n-6) \geq \lg\frac{n}{2}\ for\ n \geq 12$

$cn \lg\frac{n}{2} - c \lg 3 \cdot n - \frac{3}{2}c \lg(n-6) + \frac{3}{2}\lg 3\, c + \frac{n}{2} = cn \lg n - cn - c \lg 3 \cdot n - \frac{3}{2}c \lg(n-6) + \frac{3}{2}\lg 3\, c + \frac{n}{2} \geq cn \lg n$

If:

$-cn - c \lg 3 \cdot n - \frac{3}{2}c \lg(n-6) + \frac{3}{2}\lg 3\, c + \frac{n}{2} \geq 0$

For that we need to find the right c:

$-cn - c \lg 3 \cdot n - \frac{3}{2}c \lg(n-6) + \frac{3}{2}\lg 3\, c + \frac{n}{2} \geq 0 \Leftrightarrow$

$-cn(1 + \lg 3) - \frac{3}{2}c(\lg(n-6) - \lg 3) + \frac{n}{2} \geq 0 \Leftrightarrow$

$\frac{n}{2} \geq cn(1 + \lg 3) + \frac{3}{2}(\lg(n-6) - \lg 3) \geq cn + \underbrace{\lg(n-6) - \lg 3}_{>0 \text{ for } n \geq 12 \text{ as set before}} \geq cn \Leftrightarrow$ / divide in $n \geq 12$

$\frac{1}{2} \geq c$

Therefore, for choosing $c \leq \frac{1}{2}$ and for all $n \geq 12$ we get $T(n) = \Omega(n \lg n)$, and combining the bounds together derives

$\boxed{T(n) = \Theta(n \lg n)}$

e. $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\lg n}$

Since $a = b = 2$ then $n^{\log_2 2} = n$, which means MT cannot be applied (as there's neither equal order of growth between $n^{\log_b a}$ and $f(n)$ nor one polynomially larger than the other).

Looking at the recursion tree we find:

- At level 0: $f(n) = \frac{n}{\lg n}$
- At level 1: $2f\left(\frac{n}{2}\right) = \frac{2n}{2} / \lg\frac{n}{2} = \frac{n}{\lg n - \lg 2} = \frac{n}{\lg n - 1}$
- At level 2: $4f\left(\frac{n}{4}\right) = \frac{4n}{4} / \lg\frac{n}{4} = \frac{n}{\lg n - \lg 4} = \frac{n}{lgn - 2}$

… Until level $\lg n - 1$.

$\Rightarrow T(n) = \sum_{i=0}^{\lg n-1} \frac{n}{\lg n-i} = n\sum_{i=0}^{\lg n-1} \frac{1}{\lg n-i} \overset{(*)}{=} n\sum_{i=1}^{\lg n} \frac{1}{i} \overset{(**)}{=} \boxed{\Theta(n \lg \lg n)}$

$(*)$: Same as $(*)$ in ***b***, only with $\lg_2 n$ instead of $\lg_3 n$.

$(**)$: Same as $(**)$ in ***b***.

f. $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

After a few intuitive guesses, we will now use the substitution method to prove that $T(n) = \Theta(n)$.

For $T(n) = O(n)$ we assume that $T(k) \leq ck$ for $k \in \{1,2,\dots,n-1\}$:

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n \leq c\frac{n}{2} + c\frac{n}{4} + c\frac{n}{8} + n = \left(\frac{7c}{8} + 1\right)n \leq$ / choose $\frac{7c}{8} + 1 \leq c \Rightarrow c \geq 8$

$\leq c \cdot n$, thus $\boxed{T(n) = O(n)}$

For $T(n) = \Omega(n)$:

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n \geq n \Rightarrow \boxed{T(n) = \Omega(n)}$

since $T(n) = O(n), T(n) = \Omega(n)$ we get $\boxed{T(n) = \Theta(n)}$

g. $T(n) = T(n-1) + \frac{1}{n}$

$T(n) = T(n-1) + \frac{1}{n} = T(n-2) + \frac{1}{n-1} + \frac{1}{n} = T(n-3) + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n} = \cdots = T(2) + 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} + \frac{1}{n} =$

$T(2) + \sum_{i=1}^{n} \frac{1}{i} \underset{\substack{harmonic\ order \\ of\ growth}}{=} \Theta(1) + \Theta(\lg n) \Rightarrow \boxed{T(n) = \Theta(\lg n)}$

h. $T(n) = T(n-1) + \lg n$

$T(n) = T(n-1) + \lg n = T(n-2) + \lg(n-1) + \lg n = \cdots = T(2) + \lg 1 + \cdots + \lg(n-1) + \lg n = \sum_{i=1}^{n} \lg i + \Theta(1)$

$T(n) = O(n \lg n)$: $\sum_{i=1}^{n} \lg i \leq \sum_{i=1}^{n} \lg n = n \lg n \Rightarrow \sum_{i=1}^{n} \lg i = \boxed{O(n \lg n)}$

$T(n) = \Omega(n \lg n)$: half of the elements in the sum $\sum_{i=1}^{n} \lg i$ are $\geq \lg\left(\frac{n}{2}\right)$ so $\sum_{i=1}^{n} \lg i \geq \frac{n}{2} \lg\left(\frac{n}{2}\right) = \Theta(n \lg n) \Rightarrow$

$\sum_{i=1}^{n} \lg i = \boxed{\Omega(n \lg n)}$

$\Rightarrow \boxed{T(n) = \Theta(n \lg n)}$

i. $T(n) = T(n-2) + \frac{1}{\lg n}$

$T(n) = T(n-2) + \frac{1}{\lg n} = T(n-4) + \frac{1}{\lg(n-2)} + \frac{1}{\lg n} = \cdots = T(0) + \frac{1}{\lg 2} + \cdots + \frac{1}{\lg n} = \Theta(1) + \sum_{i=1}^{\frac{n}{2}} \frac{1}{\lg(2i)} = \Theta(1) + \sum_{i=1}^{\frac{n}{2}} \frac{1}{\lg i + 1}$

$\sum_{i=1}^{\frac{n}{2}} \frac{1}{\lg i+1} = \begin{bmatrix} \lg i + 1 = k \Rightarrow & i = 2^{k-1} \\ i = 1 \Rightarrow & k = 1 \\ i = \frac{n}{2} \Rightarrow & k = \lg\frac{n}{2} + 1 \end{bmatrix} = \sum_{k=1}^{\lg\frac{n}{2}+1} \frac{1}{k} = \Theta\left(\lg\left(\lg\frac{n}{2} + 1\right)\right) \underset{\begin{bmatrix} \lg\frac{n}{2}+1= \\ \lg n - \lg 2 + 1= \\ \lg n - 1 + 1 = \lg n \end{bmatrix}}{=} \Theta(\lg \lg n)$

$\Rightarrow \boxed{T(n) = \Theta(\lg \lg n)}$

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

First we will calculate the depth of the recursion tree, by noticing each step powers up n by $1/2$:

$$\underbrace{\left(\left(n^{1/2}\right)^{1/2}\right)^{\dots 1/2}}_{k\ times} = 2 \Rightarrow n^{\frac{1}{2^k}} = 2 \Rightarrow \log_n 2 = \frac{1}{2^k} \Rightarrow 2^k = \frac{1}{\log_n 2} = \lg n \Rightarrow k = \lg\lg n$$

Now we will look at the recursion tree and accumulate the contribution of each level:

- Level 0: $T(n) = n^{1/2}T\left(n^{1/2}\right) + \boxed{n}$ → level contribution: n
- Level 1: $n^{1/2}T\left(n^{1/2}\right) = \boxed{n^{1/2}}\left(n^{1/4}T\left(n^{1/4}\right) + \boxed{n^{1/2}}\right)$ → level contribution: $n^{1/2} \cdot n^{1/2} = n$

… (each level donates n to the total sum) Until level $\lg\lg n - 1$.

⇒ $\boxed{T(n) = \Theta(n \cdot \lg\lg n)}$

Page 109, problem 4-5:

a.

Let there be a set of chips with more than half of them bad and the others are good.

Since there are more bad chips than good, we can find a subset of bad chips of the same size as the set of the good chips, and set them to act as follows: all the good chips will be declared as bad, and all the bad chips in that subset will be declared as good (the bad chips not in that subset will be declared bad). The behavior of the remainder of the bad chips is irrelevant. This way the bad chips in this subset behave in a corresponding way to the good chips – declare good amongst themselves and bad on everyone else. Therefore the professor will not have any strategy effective to find any good chip with absolute accuracy.

b.

Given n chips of which $m > \frac{n}{2}$ chips are <good> (and the others are <bad>) with the problem of finding 1 <good> chip, here is a reduction using $\lfloor n/2 \rfloor$ pairwise comparisons to the problem of nearly half the size. Note that “good” / “bad” indicates a returned value from putting 2 chips in the test, and <good> / <bad> indicates the type of the chips.

1. Let M be an empty set for the chips to be constructed for the reduced problem.
2. Pick $\left\lfloor \frac{n}{2} \right\rfloor$ pairs (without recurrence) and compare them.
3. If n is even (all chips were compared at some point):
   - 3.1. If **exactly one** comparison result is “good-good”, pick one of the chips in that pair, **return it and finish** (as it is definitely a <good> chip).
   - 3.2. If more than one comparison return “good-good”, pick one chip of each such pair and insert it into M. All pairs that return at least one “bad” are thrown away.

4. If n is odd (**exactly** 1 chip is left out of the comparisons):
   4.1. If all comparison results return at least one "bad", **return the 1 left out of the comparisons and finish** (as it is definitely a <good> chip).
   4.2. If **exactly one** comparison result is "good-good", pick one of the chips in that pair, **return it and finish** (as it is definitely a <good> chip).
   4.3. If more than one comparison return "good-good":
      4.3.1. If the total number of pairs is even ($\frac{n-1}{2} \in \mathbb{N}_{even}$ pairs), pick one of the chips in every "good-good" pair and insert it into M. Also insert the one chip not compared into M.
      4.3.2. If the total number of pairs is odd ($\frac{n-1}{2} \in \mathbb{N}_{odd}$ pairs), pick one of the chips in every "good-good" pair and insert it into M. Do **NOT** insert the one chip not compared into M.

Correctness:

3. If n is even:
   3.1. If all comparisons **but one** returned at least one "bad", then there was a maximum pairing of <good> with <bad>. Since there are more <good> than <bad> chips and n is even, that means the pair left is necessarily <good, good>, thus picking one of them answers the problem.
   3.2. If more than one comparison return "good-good", we have no way of knowing whether these are <good, good> pairs or <bad, bad> pairs, however any mix of <good, bad> will always have at least one "bad" value (e.g. the <good> indicating about the <bad>). Thus removing all <good, bad> pairs still keeps more <good> chips than <bad> chips. Therefore we have more <good, good> pairs than <bad, bad> pairs, thus choosing one of each pair to go to M will leave M satisfying the rule of more <good> than <bad> chips. Also, $|M| \leq \frac{n}{2}$.
4. If n is odd:
   4.1. If **all** comparisons returned at least one "bad", then there was no pairing of <good, good> that can be recognized, meaning all pairs have at least one <bad>. Since there are more <good> than <bad> chips, that must mean the 1 uncompared chip is necessarily <good>, thus can be returned as answer to the problem.
   4.2. If all comparisons **but one** returned at least one "bad", then they all contain at least one <bad> chip. The "good-good" result must resulted from a <good, good> comparison, because <bad, bad> would have meant there are more <bad> than <good> chips (regardless of the type of the 1 uncompared chip). Thus returning one of the pair that returned "good-good" solves the problem.
   4.3. If there are more than one comparison that resulted with "good-good", first we notice that removing all pairs that have at least one "bad" means removing at least one <bad> (or 2 in the best case), meaning what's left – all pairs that returned "good-good" + the 1 uncompared chip – still keep more <good> than <bad> chips. Another thing to notice is that all the "good-good" chips necessarily contain either 2 <bad> chips or 2 <good> chips.
      4.3.1. If the number of "good-good" pairs is even, say $2m$, and we take the uncompared chip anyway:
         - If the uncompared chip is <good>, it could be (worst case) that the $2m$ pairs are m pairs of <good> chips + m pairs of <bad> chips, and the uncompared <good> keeps the condition of more <good> than <bad>

chips. Thus taking 1 chip of each "good-good" couple + the uncompared one results in m <bad> and $m+1$ <good> chips inserted into M, keeping the condition.

- If the uncompared chip is <bad>, given we have more <good> than <bad> chips, it must mean the $2m$ pairs divide in the worst-case into $m+1$ <good, good> and $m-1$ <bad, bad>. Thus taking 1 chip of each "good-good" pair + the uncompared one results (worst-case) in $m+1$ <good> and m <bad> chips inserted into M, keeping the condition.

4.3.2. If the number of "good-good" pairs is odd, say $2m+1$, and we don't take the uncompared chip, the case must be in the worst-case that $m+1$ of those pairs are <good, good> and m are <bad, bad>, since even if the uncompared chip was <good>, having $m+1$ <bad, bad> and m <good, good> (let alone more <bad, bad> pairs than that) would have broken the assumption of more <good> than <bad> chips – we would have had $2m+2$ <bad> and $2m+1$ <good>.

And so by taking one chip of each "good-good" pair and WITHOUT taking the uncompared chip, we satisfy at least $m+1$ <good> and at most m <bad> chips inserted into M, thus keeping the condition.

c.

First we find one good chip recursively applying the solution given in ***b***, and the recurrence equation is (dismissing floor/ceiling notations):

$$T(n) = T\left(\frac{n}{2}\right) + \frac{n}{2}$$

Case 3 of the master theorem applies here:

- $f(n) = \frac{n}{2}, n^{\lg_b a+\epsilon} = n^{\lg_2 1+\epsilon} = n^{\epsilon} \Rightarrow$ choosing $\epsilon = \frac{1}{2} > 0$ satisfies $f(n) = \Theta(n) = \Omega\left(n^{\frac{1}{2}}\right) = \Omega\left(n^{\lg_b a+\epsilon}\right)$
- $a \cdot f\left(\frac{n}{b}\right) = f\left(\frac{n}{2}\right) = \frac{n}{4} \leq c \cdot f(n) = c \cdot \frac{n}{2}$, for $c = \frac{2}{3}$, for all $n \geq 1$.

$\Rightarrow T(n) = \Theta\left(f(n)\right) = \Theta(n)$

After finding one good chip we simply apply a comparison between it and all the other $n-1$ chips, as the indication of the one good chip we have is reliable – it always tells the truth.

The total cost of the entire operation would then be: $\boxed{\Theta(n) + \Theta(n-1) = \Theta(n)}$, as required to prove.