Homework 1 Solutions
Fundamental Algorithms, Spring 2008, Professor Yap

Due: Wed Feb 6, in class.
HOMEWORK with SOLUTION, prepared by Instructor and T.A.s

INSTRUCTIONS:

- Please read questions carefully. When in doubt, please ask. You may also post general questions to the homework discussion forum in class website. Also, bring your questions to recitation on Monday.
- There are links from the homework page to the old homeworks from previous classes, including solutions. Feel free to study these.

---

1. (10 Points)
   Exercise 7.1 in Lecture 1. Assume $f(n) \geq 1$ (ev.).
   (a) Show that $f(n) = n^{\mathcal{O}(1)}$ iff there exists $k > 0$ such that $f(n) = \mathcal{O}(n^k)$. This is mainly an exercise in unraveling our notations!
   (b) Show a counter example to (a) in case $f(n) \geq 1$ (ev.) is false.

   SOLUTION: a) Let $f(n) \geq 1$ (ev) and assume that $f(n) = n^{\mathcal{O}(1)}$. Therefore there exists a $g \in \mathcal{O}(1)$ such that $g(n) \leq k$ (ev) and $f(n) \leq n^{g(n)} \leq n^k$ (ev). This shows that $f(n) \in \mathcal{O}(n^k)$. In the other direction, suppose $f = O(n^k)$. Then $f \leq Cn^k$ (ev) for some $C > 1$. Thus $f \leq n^{k+\epsilon}$ (ev) for any $\epsilon > 0$ (we This shows $f = n^{O(1)}$.
   b) To find a counterexample, let $f(n) = 1/2$. Clearly $f = \mathcal{O}(n)$. But $f(n) \neq n^{\mathcal{O}(1)}$ because if $f(n) = n^{g(n)}$ for some function $g(n)$, then clearly $g(n) < 0$ (ev). But this means $g(n) \neq \mathcal{O}(1)$. Recall that $g \in \mathcal{O}(1)$ implies $g \geq 0$ (ev).

2. (25 Points)
   Do Exercise 7.5, Lecture 1 (in 8 parts). Provide either a counter-example when false or a proof when true. The base b of logarithms is arbitrary but fixed, and $b > 1$. Assume the functions f, g are arbitrary (do not assume that f and g are ≥ 0 eventually).
   (a) $f = \mathcal{O}(g)$ implies $g = \mathcal{O}(f)$.
   (b) $\max\{f, g\} = \Theta(f + g)$.
   (c) If $g > 1$ and $f = \mathcal{O}(g)$ then $\ln f = \mathcal{O}(\ln g)$. HINT: careful!
   (d) $f = \mathcal{O}(g)$ implies $f \circ \log = \mathcal{O}(g \circ \log)$. Assume that $g \circ \log$ and $f \circ \log$ are complexity functions.
   (e) $f = \mathcal{O}(g)$ implies $2^f = \mathcal{O}(2^g)$.
   (f) $f = o(g)$ implies $2^f = \mathcal{O}(2^g)$.
   (g) $f = \mathcal{O}(f^2)$.
   (h) $f(n) = \Theta(f(n/2))$.

SOLUTION: Only two statements are true!

(a) False. $f = n$ and $g = n^2$.

(b) False. Take f any function that is eventually non-zero. Then take $g = -f$

(c) False. Take $f = 1/2$ and $g > 1$ as anything. $\log(1/2) < 0$ This solution takes advantage of the fact that if $\lg f = \mathcal{O}(\lg g)$ then $\lg f \geq 0$ (ev). HERE is another solution which does not exploit this property (and I think is more insightful): Let $f = 2$ and $g(x) = (x+1)/x = 1 + (1/x)$. Then $\ln g > 0$ but $\lg g(n) \to 0$ as $x \to \infty$. [In fact, $\lg g(x) < 1/(2x)$, but you don't need to not know this]. Clearly, $\lg f = 1$ but there is no constant $C > 0$ such that $\lg f = 1 \leq C \lg g$.

(d) True. We have $f = \mathcal{O}(g)$ implies there is some $C > 0$ and x_0 such that for all $x > x_0$, $f(x) \leq Cg(x)$. Thus, $f(\log(x)) \leq Cg(\log(x))$ for all $x \geq e^{x_0}$.

(e) False. Let $f = 2n$ and $g = n$.

(f) True. $f = o(g)$ implies that for all $C > 0$, $0 \leq f \leq C * g$ (ev). Taking $C = 1$, we obtain that $0 \leq 2^f \leq 2^g$ or in other words $2^f \in \mathcal{O}(2^g)$.

(g) False. Let $f = 1/n$.

(h) False. $f = 2^n$.

3. (10 Points)
Exercise 7.7, Lecture 1. Let $f(x) = \sin x$ and $g(x) = 1$.
(i) Prove $f \preceq g$ or its negation.
(ii) Prove $g \preceq f$ or its negation.

HINT: To prove that $f \not\preceq g$, you need to show that for *all* choices of $C > 0$ and $x_0 > 0$, some relationship between f and g fails.

SOLUTION:

(i) CLAIM: $f \preceq g$. Pf: Choose $C = 1$. Then for all $x \in \mathbb{R}$, we have $f(x) = \sin x \leq 1 = g(x)$.

(ii) Note that $f \preceq g$ fails because f is periodic. Hence we will prove the negation: CLAIM: $g \not\preceq f$. Pf: To see this, note that for all $C > 0$ and x_0, there exists $x > x_0$ such that $f(x) = 0$. Hence $g(x) \leq Cf(x)$ does not hold.

4. (10 Points)
Exercise 7.8, Lecture 1. This exercise shows three (increasingly strong) notions of lower bounds. Suppose $T_A(n)$ is the running time of an algorithm A.
(a) Suppose you have constructed an infinite sequence of inputs $I_1, I_2, \ldots$ of sizes $n_1 < n_2 < \cdots$ such that A on I_i takes time more than $f(n_i)$. How can you express this lower bound result using our asymptotic notations?
(b) In the spirit of (a), what would it take to prove a lower bound of the form $T_A(n) \neq \mathcal{O}(f(n))$? What must you show about of your constructed inputs $I_1, I_2, \ldots$.
(c) What does it take to prove a lower bound of the form $T_A(n) = \Omega(f(n))$?

SOLUTION:

(a) $T_A(n) \neq o(f(n))$. To see this, observe that there is a sequence of inputs of size $n_1, n_2, \ldots$ such that $T_A(n)/f(n) \not\longrightarrow 0$.

(b) To show that $T_A(n) \neq \mathcal{O}(f(n))$, we must show that for every $C > 0$ and for every x_0, there exists an $x > x_0$, such that $Cf < T_A(n)$.

(c) We must must show that there exists a C and n_0 such that for every $n \geq n_0$, $T_A(n) \geq Cf(n)$. NOTE: In particular, we need $n_i = n_0 + i)$ for some integer $n_0 \geq 0$. In most cases, an algorithm A only has positive integer size inputs; so $T_A(n)$ is not naturally defined if n is not-integer. In this case, we can simply define $T_A(n)$ to be $T_A(\lceil n \rceil)$.

5. (20 Points)
Exercuse A.6 in the Appendix of Lecture 1. Prove these basic facts about binary trees, assuming $n \geq 1$.
(a) A full binary tree on n leaves has $n - 1$ internal nodes.
(b) Show that every binary tree on n nodes has height at least $\lceil \lg(1 + n) \rceil - 1$. HINT: define $M(h)$ to be the maximum number of nodes in a binary tree of height h.
(c) Show that the bound in (b) is tight for each n.
(d) Show that a binary tree on $n \geq 1$ leaves has height at least $\lceil \lg n \rceil$. HINT: use a modified version of $M(h)$.
(e) Show that the bound in (d) is tight for each n.

SOLUTION: (a) Use structural induction.
BASE CASE: assume n (the number of leaves) is at least 1. Note that a full binary tree, if it has any leaves, must have at least 2 leaves. So $n = 2$ is our base case. The result is clearly true here.
INDUCTIVE CASE: assume $n > 2$. Let $n(T)$ and $i(T)$ denote the number of leaves of T and number of internal nodes of T. Thus, $n = n(T)$. Our goal is to prove

$$n(T) = 1 + i(T).$$

Note that the size of T is $|T| = n(T) + i(T)$. Let T_L, T_R be the left and right subtrees. Without loss of generality, let $|T_L| \leq |T_R|$. Assume first that $n(T_L) = 1$. Then $n(T_R) > 1$. By induction, $n(T_R) = i(T_R) + 1$. Then $n(T) = 1 + n(T_R)$ and $i(T) = 1 + i(T_R)$. This proves that

$$\begin{array}{rcll} n(T) & = & 1 + n(T_R) & \text{(by definition of } n(T)) \\ & = & 1 + (i(T_R) + 1) & \text{(by induction hypothesis)} \\ & = & 1 + i(T) & \text{(by definition of } i(T)) \end{array}$$

Finally, assume $n(T_L) > 1$. Then we have

$$\begin{array}{rcll} n(T) & = & n(T_L) + n(T_R) & \text{(by definition of } n(T)) \\ & = & (i(T_L) + 1) + (i(T_R) + 1) & \text{(by induction hypothesis)} \\ & = & 1 + i(T) & \text{(since } i(T) = 1 + i(T_L) + i(T_R)) \end{array}$$

(b) Then we have $M(h) = 1 + 2M(h-1)$ and $M(0) = 1$. Thus $M(h) = 2^{h+1} - 1$. Thus we have $n \leq M(h)$ and $n + 1 \leq 2^{h+1}$ and, by taking logarithm, $h \geq \lg(n+1) - 1$. Since h is an integer, we can take ceiling of $\lg(n + 1)$.
REMARK: this technique of defining $M(h)$ is a good general way to prove lower bound on heights. There is an analogous technique to prove upper bounds on height (see Chapter on AVL trees).
(c) We must show that for every $n \geq 1$, there are binary trees on n nodes with height $h(n) := \lceil \lg(1 + n) \rceil - 1$. If $n = 1$, $h(1) = 0$, and the result is true. Suppose $n > 1$ and n is even. Then

$$\begin{array}{rcll} h(n) & = & \lceil \lg(1 + n) \rceil - 1 & \\ & = & \lceil \lg((1 + n)/2) \rceil & \\ & = & \lceil \lg(1 + (n/2)) \rceil & \text{since } n \text{ even} \\ & = & h(n/2) + 1 & \end{array}$$

By induction hypothesis, there is a binary tree on $n/2$ nodes of height $h(n/2)$. Consider the binary tree T_n on n nodes where the left subtree has $n/2$ nodes and the right subtree has $(n/2) - 1$ nodes. The height of T_n is $1 + h(n/2)$, which is equal to $h(n)$.
What if n is odd? In this case,

$$\begin{array}{rcl} h(n) & = & \lceil \lg(1 + n) \rceil - 1 \\ & = & \lceil \lg((1 + n)/2) \rceil \\ & = & \lceil \lg(1 + (n - 1)/2)) \rceil \\ & = & h((n - 1)/2) + 1 \end{array}$$

Now construct T_n so that its left and right subtrees are both $T_{(n-1)/2}$. Hence T_n has height $1 + h((n - 1)/2)$ which is equal to $h(n)$.
(d) We can use the same method as (b), but define a modified function $M'(h)$ to be the maximum number of leaves in a binary tree of height h. Then $M'(h) = 2M'(h-1)$ where $M'(1) = 2$. Then $M'(h) = 2^h$. Hence $n \leq M'(h) = 2^h$. Hence $h \geq \lg n$, and again we can take ceiling.

6. (10 Points)
Do Exercise 11.3, Lecture 2. Order the following 5 functions in order of increasing Θ-order: (a) $\log^2 n$, (b) $n/\log^4 n$, (c) $\sqrt{n}$, (d) $n2^{-n}$, (e) $\log\log n$.

SOLUTION: The order is $(d) < (e) < (a) < (c) < (b)$.