Homework 1
Fundamental Algorithms, Spring 2008, Professor Yap

Due: Wed Feb 6, in class.

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.

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) $f(n) = \Theta(f(n/2))$.

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.

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 $T_A(n) = \Omega(f(n))$?

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.

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) $\log\log n$.