18-645/SP07: How to Write Fast Code
Assignment 1
Due Date: Thu Jan 31 6:00pm
http://www.ece.cmu.edu/∼pueschel/teaching/18-645-CMU-spring08/course.html

Submission instructions: If you have an electronic version of your assignment (preferably made using LaTeX, but other forms, including scanned copies are okay), email them to: <schellap+18645-assign1@andrew.cmu.edu>. Paper submissions need to be dropped off with one of the TAs at PH-B10 or with Carol Patterson at PH-B15. Late submissions will not be graded.

1. *(9 pts)* Show that the following identities hold by determining the explicit constants c and n_0 that are a part of the definition of O.
   (a) $n + 1 = O(n)$
   (b) $n^3 + 2n^2 + 3n + 4 = O(n^3)$
   (c) $n^5 = O(n^{\log_2 n})$

2. *(14 pts)* You know that $O(n + 1) = O(n)$. Similarly, simplify the following as much as possible and briefly justify.
   (a) $O(100)$
   (b) $O(100 + (1/n)$
   (c) $O(3n^2 + \sqrt{n})$
   (d) $O(\log_3(n))$
   (e) $O(n^{2.1} + n^2 \log(n))$
   (f) $O(mn + n)$
   (g) $O(m \log(n) + n \log(m) + n)$

3. *(9 pts)*
   (i) In class, you learned that $\Theta(log_a n) = \Theta(log_b n)$ for $a, b > 1$. Does $\Theta(a^n) = \Theta(b^n)$ hold? Justify your answer.
   (ii) Show that for $k > 0,\ \alpha > 1$: $n^k = O(\alpha^n)$ (i.e., polynomial functions grow slower than exponential functions).
   (iii) Find a function $f(n)$ such that $f(n) = O(1)$, $f(n) > 0$ for all n, and $f(n) \neq \Theta(1)$. Justify the answer.

4. *(18 pts)* Give asymptotic bounds (O, Ω, or Θ) for $T(n)$ in each of the following recurrences. Make your bounds as tight as possible. Justify your answers.
   (a) $T(n) = 2T(n/2) + n^2$.
   (b) $T(n) = T(9n/10) + n$.
   (c) $T(n) = 16T(n/4) + n^2$.
   (d) $T(n) = 7T(n/3) + n^2$.
   (e) $T(n) = 2T(n/4) + \sqrt{n}$.
   (f) $T(n) = 4T(n/2) + n^2 \log n$.

5. *(10 pts)* Solve 4(a) exactly for $T(1) = 1$ asumming $n = 2^k$.

6. *(15 pts)* Consider two polynomials: $h(x) = h_{n-1}x^{n-1} + ... + h_0$ and $p(x) = p_{n-1}x^{n-1} + ... + p_0$ of the same degree $n-1$.

   Compute the exact (arithmetic) cost

   $$C(n) = (\text{number of additions}, \text{number of multiplications})$$

   for multiplying the polynomials

   (a) by definition;

   (b) using the Karatsuba algorithm, recursively applied, assuming $n = 2^k$.

7. *(15 pts)* Solve the recurrence $f_0 = 1,\ f_1 = 1,\ f_n = f_{n-1} + 2f_{n-2}$, using the method of generating functions.

8. *(10 pts)* Consider a polynomial of the third degree: $a(x) = a_0 + xa_1 + x^2a_2 + x^3a_3$.

   (a) Compute the exact (arithmetic) cost

   $$D = (\text{number of additions}, \text{number of multiplications}):$$

   for evaluating $a(x)$ at a point $x = x_0$

   i. by definition (in a straightforward way, without using any tricks)

   ii. if the polynomial is expressed as: $a(x) = a_0 + x(a_1 + x(a_2 + xa_3))$

   (b) Now, determine the cost D of evaluating an n-th degree polynomials $p(x) = a_0 + \cdots + a_nx^n$ using the same trick as in ii. above. The method is called Horner's scheme.