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) $n^5 = O(n^{\log_2 n})$

   Solution:

   (a) $n + 1 = O(n)$

   For all $n \geq 1$,

   $$n + 1 \leq n + n = 2n.$$

   Thus, for $c = 2, n_0 = 1$, we have $n + 1 \leq cn$ for all $n \geq n_0$.

   (b) $n^3 + 2n^2 + 3n + 4 = O(n^3)$

   The trick is to get rid of all lower terms by converting them into multiples of n^3.

   For all $n \geq 1$:

   $$\begin{aligned} n^3 + 2n^2 + 3n + 4 &\leq n^3 + 2n^3 + 3n^3 + 4n^3 \\ &\leq 10n^3. \end{aligned}$$

   Therefore, we can choose $c = 10,\ n_0 = 1$ in the definition of O.

   (c) $n^5 = O(n^{\log_2 n})$

   Since $5 \leq \log_2(n)$, for $n \geq 2^5$, we have for $n \geq 32$,

   $$n^5 \leq n^{\log_2(n)}.$$

   Hence, we can choose $c = 1,\ n_0 = 32$.

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) $O(m \log(n) + n \log(m) + n)$

   Solution:

   (a) $O(100)$

   Constants don't matter, so $O(100) = O(1)$.

(b) $O(100 + (1/n))$

As n grows, $(1/n)$ goes negligible compared to 100 because $\lim_{n\to\infty} \frac{1/n}{100} = 0$. Therefore $O(100 + (1/n)) = O(1)$.

(c) $O(3n^2 + \sqrt{n})$

$\sqrt{n}$ is negligible compared to $3n^2$ because $\lim_{n\to\infty} \frac{\sqrt{n}}{3n^2} = 0$. Hence $O(3n^2 + \sqrt{n}) = O(n^2)$.

(d) $O(\log_3(n))$

As seen in class, $O(\log_3(n)) = O(\log(n))$.

(e) $O(n^{2.1} + n^2 \log(n))$

$n^2 \log(n)$ is negligible compared to $n^{2.1}$ because $\lim_{n\to\infty} \frac{n^2 \log(n)}{n^{2.1}} = 0$. Therefore $O(n^{2.1} + n^2 \log(n)) = O(n^{2.1})$.

(f) $O(mn + n)$

n is negligible compared to mn because $\lim_{m\to\infty} \frac{n}{mn} = 0$. Hence $O(mn + n) = O(mn)$.

(g) $O(m \log(n) + n \log(m) + n)$

n is negligible compared to $n \log(m)$ because $\lim_{m\to\infty} \frac{n}{n \log(m)} = 0$. $m \log(n)$ and $n \log(m)$ are not further comparable. Therefore $O(m \log(n) + n \log(m) + n) = O(m \log(n) + n \log(m))$

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.

Solution:

(i) This is true only if $a = b$. Otherwise, let's assume $a > b$. We show that $a^n \neq O(b^n)$ through a proof by contradiction:
Assume $a^n = O(b^n)$. Then, by defintion of O, there is a constant c and n_0 such that for all $n \geq n_0$:

$$a^n \leq cb^n.$$

This implies (by applying the base-a log on both sides),

$$n \leq n \log_a b + \log_a c.$$

Since $a > b$ and thus $\log_a b < 1$, we can solve for n as

$$n \leq \log_a c/(1 - \log_a b), \quad \text{for all } n \geq n_0,$$

which is a contradiction (it does not hold since n grows to infinity); thus, the original assumption is wrong and we have $a^n \neq O(b^n)$ as desired.

(ii) We consider $\lim_{x\to\infty} x^k/\alpha^x$ and apply L'Hospital's rule

http://mathworld.wolfram.com/LHospitalsRule.html

k times. Remember that the derivative of α^x is $\log_e(\alpha)\alpha^x$:

$$\lim_{x\to\infty} \frac{x^k}{\alpha^x} = \lim_{x\to\infty} \frac{kx^{k-1}}{\log_e(\alpha)\alpha^x} = ... = \lim_{x\to\infty} \frac{k!}{(\log_e(\alpha))^k \alpha^x} = 0.$$

This means, if we choose any $c > 0$, then there is an n_0 such that

$$n^k/\alpha^k < c,$$

which implies $n^k < c\alpha^k$ as desired.

(iii) $f(n) = 1/n$. Obviously $0 < 1/n < 1$ for $n \geq 1$ and thus $1/n = O(1)$. Assume now that also $1/n = \Omega(1)$ and show that this leads to a contradiction. $1/n = \Omega(1)$ means that there is a constant $c > 0$ and n_0 such that

$$c \cdot 1 \leq 1/n, \quad \text{for } n \geq n_0,$$

which is obviously wrong (whenever $n > 1/c$).

Not easy to find an algorithm with $1/n$ as cost function :-).

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) $T(n) = 4T(n/2) + n^2 \log n$.

Solution:

(a) $T(n) = 2T(n/2) + n^2$. $a = 2, b = 2, f(n) = n^2$. This is Case 3. $T(n) = \Theta(n^2)$.
(b) $T(n) = T(9n/10) + n$. $a = 1, b = 10/9, f(n) = n$. This is Case 3. $T(n) = \Theta(n)$.
(c) $T(n) = 16T(n/4) + n^2$. $a = 16, b = 4, f(n) = n^2$. This is Case 2. $T(n) = \Theta(n^2 \log n)$.
(d) $T(n) = 7T(n/3) + n^2$. $a = 7, b = 3, f(n) = n^2$. This is Case 3. $T(n) = \Theta(n^2)$.
(e) $T(n) = 2T(n/4) + \sqrt{n}$. $a = 2, b = 4, f(n) = \sqrt{n}$. This is Case 2. $T(n) = \Theta(\sqrt{n} \log n)$.
(f) $T(n) = 4T(n/2) + n^2 \log n$. This problem does not fall into any of the cases.

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

Solution: Let $n = 2^k$ and $u_k = T(2^k)$. The problem is now $u_k = 2u_{k-1} + 2^{2k}$ and $u_0 = 1$.

To get some intuition, we unwind the first terms of the sequence.

$$\begin{aligned} u_k &= 2u_{k-1} + 2^{2k} \\ &= 2(2u_{k-2} + 2^{2k-2}) + 2^{2k} \\ &= 2^2 u_{k-2} + 2^{2k} + 2^{2k-1} \\ &= 2^3 u_{k-3} + 2^{2k} + 2^{2k-1} + 2^{2k-2} \end{aligned}$$

Clearly $u_k = 2^k + 2^{2k} + \cdots + 2^{k+1}$ (proof is done by induction.).

$$\begin{aligned} u_k &= 2^k + 2^{2k} + \cdots + 2^{k+1} \\ &= 2^k + 2^{k+1}(1 + \cdots + 2^(k-1)) \\ &= 2^k + 2^{k+1}\frac{1-2^k}{1-2} \\ &= 2^k + 2^{k+1}(2^k - 1) \\ &= 2^{2k+1} - 2^k \end{aligned}$$

By returning to the standard notation, we finaly find: $T(n) = 2n^2 - n$.

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 $n = 2^k$.

   Solution:

   (a) By definition, it is necessary to multiply each factor by all others. Therefore, $C_m(n) = n^2$.
   One needs to figure out that there are as many additions as there are multiplications except that one of the multiplication goes for free in each monomials of the final polynomial(of degree $2n-2$). Therefore $C_a(n) = n^2 - 2n + 1$.

   (b) As seen in class, $C(2) = (4, 3)$ with Karatsuba's algorithms.
   We now denote $p_m(x)$ is a polynomial of degree m in the variable x. To compute the recursive Karatsuba, there are two methods that are equivalent in the number of computations. Namely $p_{2m+1}(x) = p_m(x) + x^{m+1}p_m x$ or $p_{2m+1}(x) = p_m(x^2) + xp_m x^2$. We take the first one but we would find the same results with the other one.

   $$\begin{aligned} p_{2m+1}(x)p_{2m+1}(x) &= (p_m(x) + x^{m+1}p_m x)(p_m(x) + x^{m+1}p_m x) \\ &= p_m(x)p_m(x) + x^{2m+2}p_m(x)p_m(x) + \\ &\quad x^{m+1}\left((p_m(x) + p_m(x))(p_m(x) + p_m(x)) - p_m(x)p_m(x) - p_m(x)p_m(x)\right) \end{aligned}$$

   From here, we can see that we need 3 multiplications of degree m polynomials, two addtions of degree m polynomials, two additions of degree $2m$ polynomials and $2m$ scalar additions (first term and third term don't overlap and their sum has all terms except a term in x^{2m+1}. Third term has $2m+1$ monomials, between x^{m+1} and x^{3m+1}).
   Therefore, a multiplication of two degree $2m+1$ polynomials require 3 multiplications of degree m polynomials and $2(m+1) + 2(2m+1) + 2m = 8m + 4$ scalar additions.
   Therefore, a multiplication of two degree $2^k - 1$ polynomials require 3 multiplications of degree $2^{k-1} - 1$ polynomials and $2^{k+2} - 4$ scalar additions.
   By induction, it is clear that $C_m(2^k) = 3^k$. Therefore $C_m(n) = n^{log_2(3)}$

The number of additions is a bit more tricky:

$$\begin{aligned}
C_a(2^k) &= 2^{k+2} - 4 + 3C_a(2^{k-1}) \\
&= 2^{k+2} + 3 * 2^{(k-1)+2} - (4 + 3 * 4) + 3^2 C_a(2^{k-2}) \\
&= \sum_{h=0}^{h \le k-2} (3^h 2^{2+k-h} - 3^h 4) + 3^{k-1} C_a(2^1) \\
&= \sum_{h=0}^{h \le k-2} (3^h 2^{2+k-h}) - \sum_{h=0}^{h \le k-2} (3^h 4) + 3^{k-1} * 4 \\
&= 2^{2+k} \sum_{h=0}^{h \le k-2} (3/2)^h - 4 \sum_{h=0}^{h \le k-2} (3^h) + 3^{k-1} * 4 \\
&= 2^{2+k} \frac{1 - (3/2)^{k-1}}{1 - 3/2} - 4 \frac{1 - 3^{k-1}}{1 - 3} + 3^{k-1} * 4 \\
&= 2^{3+k}((3/2)^{k-1} - 1) - 2(3^{k-1} - 1) + 3^{k-1} * 4 \\
&= 16 * 3^{k-1} - 2 * 3^{k-1} + 2 + 4 * 3^{k-1} - 2^{3+k} \\
&= 18 * 3^{k-1} - 2^{3+k} + 2 \\
&= 6 * 3^k - 8 * 2^k + 2
\end{aligned}$$

Therefore $C_a(n) = 6n^{log_2(3)} - 8n + 2$.

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.

   Solution: Our generating function is:

$$F(x) = \sum_{n \ge 0} f_n x^n$$

   Step 1:

$$\sum f_n x^n = \sum f_{n-1} x^n + 2 \sum f_{n-2} x^n$$

   Step 2:

$$\begin{aligned}
\sum_{n \ge 2} f_n x^n &= \sum_{n \ge 2} f_{n-1} x^n + 2 \sum_{n \ge 2} f_{n-2} x^n \\
F(x) &= f_0 + f_1 + \sum_{n \ge 2} f_n x^n \\
F(x) &= 1 + x + \sum_{n \ge 2} f_{n-1} x^n + 2 \sum_{n \ge 2} f_{n-2} x^n \\
F(x) &= 1 + x + x \sum_{n \ge 2} f_{n-1} x^{n-1} + 2x^2 \sum_{n \ge 2} f_{n-2} x^{n-2} \\
F(x) &= 1 + x + x \sum_{k \ge 1} f_k x^k + 2x^2 \sum_{k \ge 0} f_k x^k \\
F(x) &= 1 + x + x \sum_{k \ge 0} f_k x^k + 2x^2 \sum_{k \ge 0} f_k x^k - x f_0 x^0 \\
F(x) &= 1 + xF(x) + 2x^2 F(x)
\end{aligned}$$

   Step 3:

$$\begin{aligned}
F(x) &= 1 + xF(x) + 2x^2 F(x) \\
F(x) &= \frac{1}{(1 - x - 2x^2)} = \frac{1}{(1 + x)(1 - 2x)}
\end{aligned}$$

Step 4:

$$\begin{aligned} F(x) &= \frac{A}{(1+x)} + \frac{B}{(1-2x)} \\ A(1-2x) + B(1+x) &= 1 \\ A + B &= 1 \\ -2A + B &= 0 \\ \text{Solution } A = 1/3, B &= 2/3 \end{aligned}$$

Step 5:

$$F(x) = \frac{1}{3}\sum_{n\geq 0}(-1)^n x^n + \frac{2}{3}\sum_{n\geq 0} 2^n x^n$$

Step 6:

$$f_n = \frac{1}{3}(-1)^n + \frac{2}{3}2^n = \frac{1}{3}(2^{n+1} + (-1)^n)$$

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 $p(x) = a_0 + \cdots + a_n x^n$ using the same trick as in ii. above. The method is called Horner's scheme.

Solution:

(a) i. By definition, there is 3 additions and 6 multiplications (without reusing already computed powers of x) or 3 additions and 5 multiplications (with reuse of powers of x)

ii. With this expression it goes down to 3 additions and 3 multiplications.

(b) By induction, Horner's scheme evaluates a polynom of degree n in n additions and n multiplications.