**263-2300-00: How To Write Fast Numerical Code**
Solution Assignment 1
Due Date: Thu March 10 17:00
http://www.inf.ethz.ch/personal/markusp/teaching/263-2300-ETH-spring11/course.html

1. (30pts) Solve the recurrence $g_1 = 10,\ g_2 = 6$,

$$g_n = 2 * g_{n/2} + 3 * g_{n/4}, \quad n = 2^k,\ k \geq 2.$$

Solving means determining a closed form for $g_n$.

**Solution**:

We substitute $n = 2^k$ and $g_n = g_{2^k} = f_k$ and get $f_0 = 10,\ f_1 = 6$ and

$$f_k = 2 * f_{k-1} + 3 * f_{k-2} \tag{1}$$

The generating function for $f_k$, $k \geq 0$ is

$$F(x) = \sum_{k=0}^{\infty} f_k * x^k \tag{2}$$

Now we multiply (1) by $x^k$ and sum up from $k = 2$:

$$\sum_{k=2}^{\infty} f_k * x^k = 2 * \sum_{k=2}^{\infty} f_{k-1} * x^k + 3 \sum_{k=2}^{\infty} f_{k-2} * x^k \tag{3}$$

The following holds:

$$\sum_{k=2}^{\infty} f_k * x^k = F(x) - f_0 - f_1 * x \tag{4}$$

$$\sum_{k=2}^{\infty} f_{k-1} * x^k = x(F(x) - f_0) * x \tag{5}$$

$$\sum_{k=2}^{\infty} f_{k-2} * x^k = x^2 * F(x) \tag{6}$$

Substituting (4), (5) and (6) in (3) and yields

$$F(x) - f_0 - f_1 * x = 2(x(F(x) - f_0)) + 3x^2 F(x), \tag{7}$$

and hence

$$F(x) = \frac{f_0 + (f_1 - 2f_0)x}{1 - 2x - 3x^2} \tag{8}$$

Plugging in the inital values,

$$F(x) = \frac{14x - 10}{1 - 2x - 3x^2} \tag{9}$$

Now we do PFE (partial fraction expansion):

$$F(x) = \frac{14x - 10}{(1 - 3x)(1 + x)} \tag{10}$$

$$F(x) = \frac{A}{1 - 3x} + \frac{B}{1 + x} \tag{11}$$

Using the formula from class, we get the values $A = 4$ and $B = 6$. (Alternative: at this point we know that $f_k = A3^k + B(-1)^k$; this means one can get $A, B$ by inserting the two initial values to obtain a two linear equations in two unknowns; this method is more work though.)

Expanding $F(x)$ back into a series yields:

$$F(x) = 4\sum_{k=0}^{\infty} 3^k x^k + 6\sum_{k=0}^{\infty}(-1)^k x^k \tag{12}$$

From what we read off

$$F_k = 4 * 3^k + 6 * (-1)^k \tag{13}$$

Translating back into exponential form yields the final result

$$\begin{aligned} g_n &= 4 * 3^{\log_2(n)} + 6 * (-1)^{\log_2(n)} \\ &= 4 * n^{\log_2(3)} + 6 * (-1)^{\log_2(n)} \end{aligned}$$

2. (20pts) Proof that $f_k = a^k * c + \sum_{i=0}^{k-1} a^i * s_{k-i}$ solves the recurrence $f_0 = c$, $f_k = a * f_{k-1} + s_k$, $k \geq 1$.

**Simplest solution:** Just check that the formula satisfies the recurrence.

Initial condition: $f_0 = a^0 * c = c$ as desired.

Recurrence:

$$\begin{aligned} a * f_{k-1} + s_k &= a(a^{k-1} * c + \sum_{i=0}^{k-2} a^i * s_{k-1-i}) + s_k \\ &= a^k * c + (a\sum_{i=0}^{k-2} a^i * s_{k-1-i}) + s_k \\ &= a^k * c + (\sum_{i=0}^{k-2} a^{i+1} * s_{k-1-i}) + s_k \\ &= a^k * c + (\sum_{i=0+1}^{k-2+1} a^{i+1-1} * s_{k-1-(i-1)}) + s_k \\ &= a^k * c + (\sum_{i=1}^{k-1} a^i * s_{k-i}) + s_k \\ &= a^k * c + (\sum_{i=0}^{k-1} a^i * s_{k-i}) - a^0 * s_{k-0} + s_k \\ &= a^k * c + \sum_{i=0}^{k-1} a^i * s_{k-i} \\ &= f_k \end{aligned}$$

as desired.

**Solution by induction**:

We prove by induction therefore first proving the initial element

$$a * f_{k-1} + s_k = a^k * c + \sum_{i=0}^{k-1} a^i * s_{k-i}$$

with $f_0 = c$ and $k = 1$ yields

$$a * c + s_1 = a^1 * c + \sum_{i=0}^{0} a^0 * s_1$$

We then move on to proof that it holds for any element via

$$\begin{aligned}
a * f_k + s_{k+1} &= a^{(k+1)} * c + \sum_{i=0}^{k} a^i * s_{k+1-i} \\
&= a * [a^k * c + \sum_{i=0}^{k-1} a^i * s_{k-i}] + s_{k+1} \\
&= a^{(k+1)} * c + \sum_{i=0}^{k} a^i * s_{k+1-i} \\
&= a^{k+1} * c + a * \sum_{i=0}^{k-1} a^i * s_{k-i} + s_{k+1} \\
&= a^{k+1} * c + a * \sum_{i=1}^{k+1-1} a^{i-1} * s_{k-(i-1)} + s_{k+1} \\
&= a^{k+1} * c + \sum_{i=1}^{k+1-1} a^i * s_{k-(i-1)} + s_{k+1} \\
&= a^{k+1} * c + \sum_{i=0}^{k} a^i * s_{k-(i-1)} - a^0 * s_{k-(0-1)} + s_{k+1} \\
a * f_k + s_{k+1} &= a^{k+1} * c + \sum_{i=0}^{k} a^i * s_{(k+1)-i}
\end{aligned}$$

3. (20pts) You know that $O(n+1) = O(n)$. Similarly, simplify the following as much as possible and briefly justify.

   (a) $O(2^{n^2+1})$

   (b) $O(2^{n^2+n+1})$

   (c) $O(1.01^n + n^5)$

   (d) $O(n^2 m + n\log(n) + m\log(m))$

   (e) $O(2^{n+\log_2(n)})$

   **Solution**:

   (a) $O(2^{n^2+1}) = O(2^{n^2})$, cause in $2^{n^2} * 2$ the 2 is a constant factor that can be removed

   (b) $O(2^{n^2+n+1}) = O(2^{n^2+n})$, same logic as above - just that you can not remove the $2^n$ as its not constant

   (c) $O(1.01^n + n^5) = O(1.01^n)$, $n^5$ is $O(1.01^n)$ because $\lim_{n\to\infty} \frac{n^5}{1.01^n} = 0$

   (d) $O(n^2 m + n\log(n) + m\log(m)) = O(n^2 m + m\log(m))$ We can remove $n\log(n)$ cause it will be always dominated by $n^2$, while we cannot remove $m\log(m)$ since it is not comparable to either of the other terms.

   (e) $O(2^{n+\log_2(n)}) = O(2^n * 2^{\log(n)}) = O(n2^n)$, similar to (b) we cannot remove any term, only modify it as shown.

4. (30pts) The Strassen algorithm (see http://en.wikipedia.org/wiki/Strassen_algorithm), named after Volker Strassen, showed for the first time that the standard approach for square matrix multiplication, which requires $\Theta(n^3)$ many operations, is not optimal. It works as follows.

We assume for this exercise $n = 2^k$ and that $A, B, C$ are all $n \times n$. Strassen's algorithm for computing $C = AB$ partitions the matrices into blocks of half the size:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \; B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \; C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Then first the following seven intermediate matrices are computed:
$M_1 = (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2})$
$M_2 = (A_{2,1} + A_{2,2})(B_{1,1})$
$M_3 = A_{1,1}(B_{1,2} - B_{2,2})$
$M_4 = A_{2,2}(B_{2,1} - B_{1,1})$
$M_5 = (A_{1,1} + A_{1,2})B_{2,2}$
$M_6 = (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2})$
$M_7 = (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})$

and from these the four blocks of $C$, and hence $C$, as

$C_{1,1} = M_1 + M_4 - M_5 + M_7$
$C_{1,2} = M_3 + M_5$
$C_{2,1} = M_2 + M_4$
$C_{2,2} = M_1 - M_2 + M_3 + M_6$

Answer the following:

(a) The above shows that the algorithm decomposes matrix multiplication into $u$ matrix multiplications of half the size and $v$ matrix additions of half the size. What is $u$ and $v$?

(b) We define the cost measure $C(n) = (A(n), M(n))$, where $A(n)$ is the number of (scalar) additions and $M(n)$ the number of (scalar) multiplications required for matrix multiplication. First determine recursive formulas for $A(n)$ and $M(n)$ for Strassen's algorithm. Second, solve these to get the exact addition and multiplication count if Strassen's algorithm is applied recursively for all occurring matrix multiplications. Show your work.

**Solution**:

(a) $u = 7$ and $v = 18$. A straight forward solution would require 8 matrix multiplications of half the size.

(b) Every matrix multiplication is divided into in 7 matrix multiplications and 18 matrix additions (both of half the size). For matrices of size $1 \times 1$, 1 multiplication and no addition is required. Therefore the recurrence for the number of scalar multiplications is

$$M(n) = 7 * M(n/2), \quad M(1) = 1.$$

Since the 18 matrix additions of half the size require $18(n/2)^2 = (9/2)n^2$ additions, the recurrence for the number of additions is

$$A(n) = 7 * A(n/2) + \frac{9}{2}n^2, \quad A(1) = 0.$$

As usual, we first translate the recurrences by substituting $n = 2^k$, $m(k) = M(n)$, and $a(k) = A(n)$:

$$\begin{aligned} m(k) &= 7m(k-1), \quad m(0) = 1 \\ a(k) &= 7a(k-1) + \frac{9}{2}4^k, \quad a(0) = 0. \end{aligned}$$

Now we use the formula proven in task 2 of this exercise sheet. This gives us for the multiplications:

$$m(k) = 7^k * 1 + \sum_{i=0}^{k-1} 7^i * 0 = 7^k$$

$$M(n) = 7^{\log_2(n)} = n^{log_2(7)}$$

For the additions:

$$\begin{aligned} a(k) &= 7^k * 0 + \sum_{i=0}^{k-1} 7^i * \frac{9}{2} * (4^{k-i}) \\ &= \frac{9}{2} * 4^k \sum_{i=0}^{k-1} (\frac{7}{4})^i \\ &= \frac{9}{2} * 4^k (\frac{\frac{7}{4}^k - 1}{\frac{7}{4} - 1}) \\ &= 6(7^k - 4^k) \end{aligned}$$

$$A(n) = 6(7^{\log_2(n)} - 4^{\log_2(n)}) = 6n^{\log_2(7)} - 6n^2$$

Total operations count is therefore

$$C(n) = 7n^{\log_2(7)} - 6n^2 = O(n^{\log_2(7)})$$

Attached a small table with values calculated for the number of operations.

| n | #Muls | #Adds |
|---|---|---|
| 2 | 7 | 18 |
| 4 | 49 | 198 |
| 8 | 343 | 1674 |
| 16 | 2401 | 12870 |
| 32 | 16807 | 94698 |