263-2300-00: How To Write Fast Numerical Code
Assignment 1
Due Date: Thu March 10 17:00
http://www.inf.ethz.ch/personal/markusp/teaching/263-2300-ETH-spring11/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: <s11-fastcode@inf.ethz.ch>. Paper submissions need to be dropped off at RZ H22.
Late policy: you have 3 late days, but can use at most 2 on one homework. Late submissions have to be emailed to the address above.

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.

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

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

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.