CS 667 : Homework 3(Due: Nov 4, 2009)

Problem 1. (25 points)

(a) We have 27 coins all of the same weight except one that is fake and weighs LESS than the other coins. We also have a balance scale; any number of coins can be put on one or the other side of the scale at any one time and the scale will tell us whether the two sides weigh the same or which side weighs more (or less). Can you find the fake coin with only 3 weighings? Explain (and justify your answer).

(b) We have 4 coins all of the same weight except one that is fake and may weigh LESS or MORE than the other coins (we don't know what the case is). We also have a balance scale; any number of coins can be put on one or the other side of the scale at any one time and the scale will tell us whether the two sides weigh the same or which side weighs more (or less). If two coins are weighed (one on each side of the scale) and they do not weigh the same, we can't decide by this weighing alone which of the two is the fake.

Can you find the fake coin with only 2 weighings? Explain (and justify your answer).

Problem 2. (25 points)

There are two types of NJIT professors: `nice` that give 100% of their students an `A` and `mean` that gives 80% of their students an `A` and 20% a `B+`. You need to determine whether professor `I.M.Cute` is `nice` or `mean`. Towards this you have access to his grading this past semester in the form of array $A[1...n]$, $n > 25$. An entry $A[i]$, $1 \leq i \leq n$, can be an `A` or a `B+`. However reading entry $A[i]$ for some arbitrary i, is going to cost you \$1. If your budget is only \$20, can you determine whether `I.M.Cute` is `nice` or `mean` with 90% or more confidence and at a cost of no more than \$20? Explain.

Disclaimer. Do not take seriously the definition of a `mean` NJIT Professor :-) I am neither `nice` nor `mean`!

Problem 3. (50 points)

Perfect Power Revisited. Show how you can determine that $n = a^b$ is a perfect power or not in $O(\lg^k n)$ bit operations. Keep k (strictly) smaller than 4. Note that you count bit operations now and the only input you are given is n in binary.

Problem 4. (50 points)

(a) Let $M(n)$ be the time to multiply two $n \times n$ matrices and let $S(n)$ be the time to square an $n \times n$ matrix. Show that multiplying and squaring have essentially the same difficulty: i.e. an $M(n)$ matrix multiplication algorithms implies an $O(M(n))$ squaring algorithm and and an $S(n)$ squaring algorithms implies an $O(S(n))$ matrix multiplication algorithm.

(b)

Solve the recurrence for the number of operations in Strassen's method exactly, i.e. solve

$$M(n) = 7M(n/2) + (18/4)n^2.$$

What is the base case? Explain. How many operations are required for the base case? Explain. Does it matter whether one chooses the base case to be $n = 1$ or $n = 2$?

What happens for the questions above if one uses the modification of the textbook with the recurrence becoming

$$M(n) = 7M(n/2) + (15/4)n^2.$$

Problem 5. (50 POINTS)

(a) Let n be a power of two. Show how to construct an n-input and an n-output comparison network of depth $\lg n$ in which the top output wire always carries the minimum input value and the bottom output wire always carries the maximum input value. What is the size of the network in term of comparators? Given an exact answer. Show that it can be done in $o(n \lg n)$ comparators.

(b) Show that any sorting network must have $\Omega(\lg n)$ depth, and $\Omega(n \lg n)$ comparators.

(c) Then show that any sorting network must have depth at least $\lg n$.

Note: In (a) we don't care about constants; in (b) we do care.

Problem 6. (50 POINTS)

(a) Show that an n-input sorting network must contain at least one comparator between the i-th and the $i+1$-st wire for all i such that $1 \leq i \leq n-1$.

(b) Draw the sorting network derived from selection sort and determine its depth. Justify you answer. Assume that selection sort finds in the i-th iteration the maximum of the remaining $n-i+1$ keys and moves that key in the $n-i+1$ position in the output sequence.

Both algorithms are discussed on pages 27 and 38 (exercises) of the CLRS textbook.

(c) Show that there are $n^2 - n + 2$ bitonic sequences of 0's and 1's.

Problem 7. (100 POINTS)

```
// n can be any integer dimension ; i.e. you have to take care and make it
//   a power of two if necessary
// *A, *B, *C are one dimensional arrays of n*n elements (floats)
// A[j*n+i] is the i-th row and j-th column element of a two dimensional array
// For Java use one dimensional arrays
Strassen(float *A, float *B, float *C, int n)  //Does Strassen for arbitrary n
ReadMatrix(float **A,int *n, file input-file); //Allocates space for A and reads A
SetMatrix(float *A,float *B,int n); //Allocates space for A and reads A
PrintMatrix(float *A,int n,  file output-file,)//Prints A into file output-file
```

You need to implement the following interface

```
% ./strassen  input-A input-B output-C
or
% java strassen input-A input-B output-C
```

where `input-A, input-B` are files containing input matrices A, B and `output-C` is the file that will contain the output $A \times B$ of Strassen's method. All three files have the same format. The first line contains the dimension n in the form of an integer. Subsequent lines contain in the form of floats the input elements in row-major format. That is the first $n = 5$ values `1.0 2.0 3.0 4.0 5.0` are the elements of the first row of the 5×5 array. The next 5 values `6.0 7.0 8.0 9.0 and 10.0` are the elements of the second row and so on. Files `input-A, input-B` are read through `ReadMatrix` and file `output-C` is written by `PrintMatrix`. `SetMatrix` allows one to set copy B into A internally.

```
5
1.0 2.0 3.0 4.0 5.0
6.0 7.0
     8.0 9.0 10.0
1.0
  2.0 3.0 4.0 5.0
1.0
  2.0 3.0 4.0 5.0
1.0 2.0 3.0 4.0 5.0
```

Note. The input array(s) can be of dimension say 17×17. After reading such matrices it's up to you to decide how to store such a matrix; Strassen (textbook description) can only deal with 16×16 or 32×32 matrices but not a 17×17 one. When you print the results into `output-C` make sure it is that of a 17×17 matrix and not that of a matrix of some other dimension. For other assumptions, deviations or instructions, provide a `readme.txt` file with your code.

Problem 8. (100 POINTS)

Implement the algorithm for inversion based on Schur decomposition (Subject 3, pages 13-15). Your algorithm should work with any dimension n input matrices A. Adjustments to the dimension should be internal; if the input is of dimension n so should the output even if internally you are using a dimension higher than n. The three functions of the previous problem `ReadMatrix, SetMatrix, PrintMatrix` can/must be reused for I/O. In order to avoid (and be able to deal with) problems with singularities you may wish to read the last page of the Subject 3 notes.

```
 // n can be any integer dimension ; i.e. you have to take care and make it
 //   a power of two if necessary
 // *A, *B, *C are one dimensional arrays of n*n elements (floats)
 // A[j*n+i] is the i-th row and j-th column element of a two dimensional array
 // For Java use one dimensional arrays
 RecursiveInverse(float *A, float *B, int n); //Find B=A**-1 Inverse per Sub
3/pages 14-15
 MatrixMultiply(float *A, float *B, float *C, int n);  // C= A*B
 ReadMatrix(float **A,int n, file input-file); //Allocates space for A and reads A
 SetMatrix(float *A,float *B,int n); //Allocates space for A and reads A
 PrintMatrix(float *A,int n,  file output-file,)//Prints A into file output-file

 A matrix of the following form might be used as input for testing purposes.
 Test Input Matrix ( float *mat )
   for(j=0;j<n;j++)
     for(i=0;i<n;i++) {
        if (i>j) mat[j*n+i] = (float) 0.5*i+1.0;
            else mat[j*n+i] = (float) 0.5*j+0.5;
   }
```

You need to implement the following interface

```
% ./reinverse input-A output-B
or
% java reinverse input-A output-B
```

where `input-A, output-B` are files containing input/output matrices A, B. All have the same format. For other assumptions, deviations or instructions, provide a `readme.txt` file with your code; none of the assumptions/deviations however should restrict the generality of the problem.

Testing matrices that you might decide to use include the following.

- $a_{ii} = 1$ if $i \neq n$, $a_{ij} = 0$ if $i < j < n$, and $a_{ij} = (-1)^{i+j-1}$ if $i > j$ or $j = n$.
- $b_{ij} = |i - j|$.
- $c_{ij} = n - |i - j|$.
- $d_{ij} = \max\{i, j\}|$.
- $f_{ii} = 1$ and other elements are $f_{ij} = 1/n$.