Problem Set 4: Due by November 13, 2006

Problem 1. (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. Do the same for bubble-sort as well. The algorithms are the ones of the textbook. Both algorithms are discussed on pages 27 and 38 (exercises) of the CLRS textbook.

(c) How many bitonic sequences of length n of 0s and 1s do we have? Give an exact answer.

Problem 2. (40 points)

You are given three polynomials $f(x), g(x), h(x)$ of degree bounds $n+1$, $n+1$, and $2n+1$ respectively. We would like to verify whether $f(x)g(x) = h(x)$ very fast but not with absolute certainty. The verification algorithm does not need to be deterministic; it can be a randomized algorithm.

One naive approach is to find the product $f(x)g(x)$ and compare it to $h(x)$. Finding the product takes $O(n^2)$ time (or $O(n \lg n)$ using FFT-based methods that we are going to discuss in due time) and comparison of $f(x)g(x)$ and $h(x)$ takes $O(n)$ time. The total amount of time is, however, $\omega(n)$.

Give a randomized algorithm that determines in $O(n)$ time whether $h(x)$ is $f(x)g(x)$ with confidence at least $1 - 1/n^{10}$. In addition the number of random bits you use should not exceed $O(\lg n)$. For example if you want to draw a random number in the interval $[1, N]$ it will cost you $\lg N$ bits because the largest such number requires about $\lg N$ bits to be represented.

Problem 3. (40 points)

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.

Problem 4. (40 points)

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

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

Problem 5. (40 points)

Find the page rank of the graphs of Figures 1 and 2 below. Initial values are $1/N$. Also use $0.15/N$ in the page rank formula. Iterate as many times as needed for the error to be less than 10^{-6}. Use the synchronous version as shown below.

```
   PageRank(G,V,E)
1. for all vertices u in V     /* Initialization Step */
2.    Src[u] = 1/N;
3. small = something-small;
4. while (convergence-distance > small) {
5.   for all v in V
6.     D[v]=0;
7.   for(i=0;i<|V|;i++) {
8.     Read-Adjacency-List(u,m,d1,d2,...,dm);
9.     for(j=1;j<=m;j++)
10.      D[dj] = D[dj] + Src[u]/m
11.  }
12.  for all v in V
13.    D[v] = d * D[v] + (1-d)/|V|
14.  convergence-distance = ||S-D|| /* Euclidean distance */
15.  Src=D;
16. }
```

Problem 6. (40 points)

Find the hub/authority rank of the graphs of Figures 1 and 2 below. Initial values are $1/N$. Repeat as many time as long as error is less than 10^{-6}. Use also the synchronous version of the algorithm.

Problem P1. (60 points)

Huffman coding. Says all. Arguments ($N \leq 10$) in the command-line won't exceed 10 in the file option case, it's one in the directory argument case.

```
% ./huffman-encode file1 file2 ... fileN
% ./huffman-encode dir1
% ./huffman-decode file1 file2 ... fileN
% ./huffman-decode dir1
 // Encode : converts file1 ---> file1.huf
 // Decode : From command-line file1 reads (if it exists) file1.huf and converts it into file1
 // file1 ... fileN may have suffixes: eg myfile.pdf ---> myfile.pdf.huf --> myfile.pdf
 // Operation is destructive! myfile.pdf would be erased after creating myfile.pdf.huf
```

Problem P2. (60 points)

Comparison Network Analyzer. Input file looks like

```
3 3
1 2
2 3
1 2
```

The first line contains the number of input and output lines first and then the number of comparators of the network. Afterwards the comparison network is described, one comparator per line. Input/Output lines are integers starting with 1. You need to implement code that checks whether the comparison network is a sorting network or not. If you decide it is not, you must report the inputs that causes it to fail to be a sorting network in an output file. The format of that output file is one of the two (successful and unsuccessful case) shown below.

```
Network given by (see below) is a sorting network.
3 3
1 2
2 3
1 2
```

or

```
Network given by (see below) is  NOT a sorting network.
  1:1, 2:1, 3:0 ---> 1:1, 2:0, 3:1 which is not sorted.
3 3
1 2
2 3
1 3
```

The format $a : b$ shows that the a-th input/output wire carries b. You need to implement a program that responds successfully (after compilation) to

```
./sortnet input-file result-file
or
java sortnet input-file result-file
```

Files `input-file` and `result-file` can be arbitrary file-names.
The minimum implementation is that of a function

```
sorting-network(string input-file, string output-file);
// The two input parameters are strings; you must check whether the files exists
// open them, read them or write into them as needed.
```

Note for both P1 and P2: Your code must compile under g++ or gcc versions 3.2.3 or later and Java version 1.4 or later.

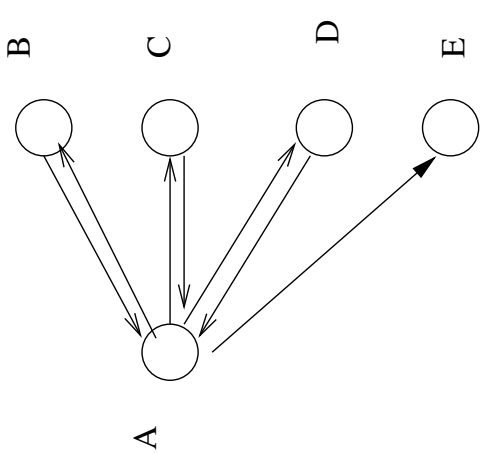

Figure 1.

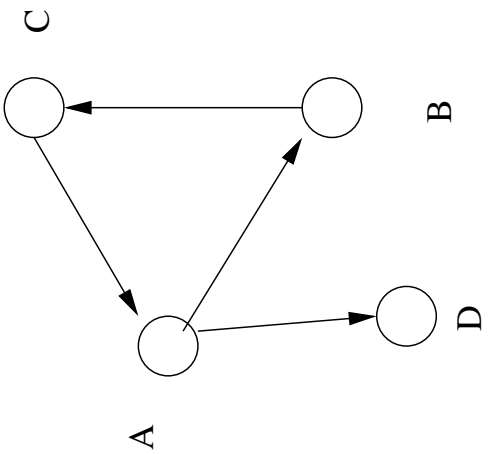

Figure 2.