L A S T H O M E W O R K

CS 667 : Homework 5(Due: Apr 17, 2012)

This Homework is Problems 1-7 and worth 250 points. You can replace some of these points with Problem 8 or 9, but you can only submit 250 points worth of problems.

Problem 1. (50 points)
(a) Give a parallel algorithm that computes F_n, the n-th Fibonacci number number in $O(\lg n)$ time (WORD model). Give parallel running time, processor size, work and actual work performed (i.e. T, P, W, W_a). Is your algorithm optimal? Explain. (Definitions for Fibonacci numbers are in Homework 1, Problem 3.)
(b) You are given the recurrence $X_i = aX_{i-1} + b$, with $X_0 = c$, where a, b, c are values stored in memory locations $M[0]...M[2]$ of the shared memory respectively. Give an efficient parallel algorithm that finds X_n (show P, T, W, W_a.)

Problem 2. (25 points)
(a) Show that any sorting network must have $\Omega(\lg n)$ depth, and $\Omega(n \lg n)$ comparators.
(b) Then show that any soring network must have depth at least $\lg n$.

Problem 3. (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. How many comparators does it use to sort n keys? (Exact values please.)

Problem 4. (50 points)
(a) Let $S = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ be a sequence of n distinct keys. The rank of x_j in the sequence S or $r(x_j, S)$ is the number of keys less than x_j in S. The problem of sorting is equivalently the problem of determining the rank of each one of the n input keys.
Determine the rank of all keys in S in $O(\lg n)$ time with a CRCW PRAM. How many processors did you use?
Sort the n keys in the same time with a CRCW PRAM. How many processors did you use?
(b) Can you repeat the two questions above for an EREW PRAM? How would the answers change? Explain.

Problem 5. (25 points)
Suppose you have a CRCW PRAM and n keys. Find the MIN in $O(1)$ time with $O(n^{1+1/7})$ processors.

Problem 6. (25 points)
Kleinberg. Find the hub/authority rank of the graph of Figure 1. Initial values are $1/N$. Iterate as many times as needed for the error to be less than 10^{-4}. (Do not forget scaling.)

Problem 7. (25 points)
PageRank. Find the page rank of the graph of Figure 1. Initial values are $1/N$. Iterate as many times as needed for the error to be less than 10^{-4}.

Problem 8. (50 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 of two numbers) and then the number of comparators of the network (second of two numbers). This second number is also the number of lines to follow after the first line. Afterwards the comparison network is described, one comparator per line. Input/Output lines are integers starting with 1. Order matters and repetitions might occur. 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 input that cause it to fail to be a sorting network in an output file. (One counterexample suffices.) The format of that output file is one of the two (successful and unsuccessful case) shown below.

```
Network shown below is a sorting network.
3 3
1 2
2 3
1 2

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 inputfile resultfile
or
java sortnet inputfile resultfile
```

Files `inputfile` and `resultfile` are arbitrary file-names. The minimum implementation is that of a function `sorting-network(string input-file, string output-file)`.

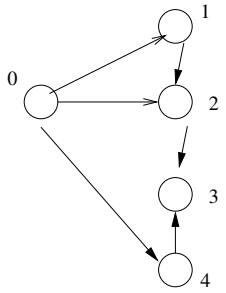

Figure 1: Problem 7 figure

Problem 9. (50 POINTS)

Implement the HITS and PageRank algorithms. The inputs will be graphs represented through an adjacency list. The command-line interface would be as follows.

```
% ./rank    rank   InitialValue Iterations InputFile
% java rank rank   InitialValue Iterations InputFile
```

The command-line parameter `rank` takes one of two values: `0` if Kleinberg's HITS is used (with the scaling as otherwise shown on page XX of Subject YY) and `1` if the Brin and Page's PageRank algorithm is used (as shown on page ZZ of Subject YY). The second parameter `InitialValue` indicates how the initial values for the ranks will be computed. If it is `0` all ranks are initialized to 0, if it is `1` they are initialized to 1. If it is `2` they are initialized to $1/N$, where N is the number of web-pages (size of the graph.) If the value is a numeric integer value other than 0,1,2 then the ranks are initialized as `InitialValue` divided by 100. Thus an `InitialValue` equal to 50, initializes all ranks to $50/100 = 0.5$. Parameter `Iterations` runs the algorithms for that number of iterations. Parameter `InputFile` describes the input graph and it has the following form. The first line contains two numbers: the number of vertices (in the example below, this is equal to five) and the number of edges that follow on separate lines (i.e. six). In each line an edge (i, j) is presented by `i j`. The graph used in class in a lecture will be represented as follows. (Note that the graphs in class have vertices in the range $1..n$, whereas in this implementation, it is $0..n-1$.

```
4 4
0 2
0 3
1 0
2 1
```

```
Kleinberg  might report, at the  14-th iteration, Authority/Hub pair values of

Base   :  0 :A/H[ 0]=0.25000/0.25000 A/H[ 1]=0.25000/0.25000 A/H[ 2]=0.25000/0.25000 A/H[ 3]=0.25000/0.25000
Iterat :  1 :A/H[ 0]=0.50000/0.81650 A/H[ 1]=0.50000/0.40825 A/H[ 2]=0.50000/0.40825 A/H[ 3]=0.50000/0.00000
Iterat :  2 :A/H[ 0]=0.31623/0.94281 A/H[ 1]=0.31623/0.23570 A/H[ 2]=0.63246/0.23570 A/H[ 3]=0.63246/0.00000
Iterat :  3 :A/H[ 0]=0.17150/0.98473 A/H[ 1]=0.17150/0.12309 A/H[ 2]=0.68599/0.12309 A/H[ 3]=0.68599/0.00000
Iterat :  4 :A/H[ 0]=0.08771/0.99612 A/H[ 1]=0.08771/0.06226 A/H[ 2]=0.70165/0.06226 A/H[ 3]=0.70165/0.00000
Iterat :  5 :A/H[ 0]=0.04411/0.99902 A/H[ 1]=0.04411/0.03122 A/H[ 2]=0.70573/0.03122 A/H[ 3]=0.70573/0.00000
Iterat :  6 :A/H[ 0]=0.02209/0.99976 A/H[ 1]=0.02209/0.01562 A/H[ 2]=0.70676/0.01562 A/H[ 3]=0.70676/0.00000
Iterat :  7 :A/H[ 0]=0.01105/0.99994 A/H[ 1]=0.01105/0.00781 A/H[ 2]=0.70702/0.00781 A/H[ 3]=0.70702/0.00000
Iterat :  8 :A/H[ 0]=0.00552/0.99998 A/H[ 1]=0.00552/0.00391 A/H[ 2]=0.70709/0.00391 A/H[ 3]=0.70709/0.00000
Iterat :  9 :A/H[ 0]=0.00276/1.00000 A/H[ 1]=0.00276/0.00195 A/H[ 2]=0.70710/0.00195 A/H[ 3]=0.70710/0.00000
Iterat : 10 :A/H[ 0]=0.00138/1.00000 A/H[ 1]=0.00138/0.00098 A/H[ 2]=0.70711/0.00098 A/H[ 3]=0.70711/0.00000
Iterat : 11 :A/H[ 0]=0.00069/1.00000 A/H[ 1]=0.00069/0.00049 A/H[ 2]=0.70711/0.00049 A/H[ 3]=0.70711/0.00000
Iterat : 12 :A/H[ 0]=0.00035/1.00000 A/H[ 1]=0.00035/0.00024 A/H[ 2]=0.70711/0.00024 A/H[ 3]=0.70711/0.00000
Iterat : 13 :A/H[ 0]=0.00017/1.00000 A/H[ 1]=0.00017/0.00012 A/H[ 2]=0.70711/0.00012 A/H[ 3]=0.70711/0.00000
Iterat : 14 :A/H[ 0]=0.00009/1.00000 A/H[ 1]=0.00009/0.00006 A/H[ 2]=0.70711/0.00006 A/H[ 3]=0.70711/0.00000

and PageRank
Base   :  0 :P[ 0]=0.25000 P[ 1]=0.25000 P[ 2]=0.25000 P[ 3]=0.25000
Iter   :  1 :P[ 0]=0.25000 P[ 1]=0.25000 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  2 :P[ 0]=0.25000 P[ 1]=0.15969 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  3 :P[ 0]=0.17323 P[ 1]=0.15969 P[ 2]=0.14375 P[ 3]=0.14375
Iter   :  4 :P[ 0]=0.17323 P[ 1]=0.15969 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  5 :P[ 0]=0.17323 P[ 1]=0.13196 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  6 :P[ 0]=0.14966 P[ 1]=0.13196 P[ 2]=0.11112 P[ 3]=0.11112
Iter   :  7 :P[ 0]=0.14966 P[ 1]=0.13196 P[ 2]=0.10111 P[ 3]=0.10111
Iter   :  8 :P[ 0]=0.14966 P[ 1]=0.12344 P[ 2]=0.10111 P[ 3]=0.10111
Iter   :  9 :P[ 0]=0.14242 P[ 1]=0.12344 P[ 2]=0.10111 P[ 3]=0.10111
Iter   : 10 :P[ 0]=0.14242 P[ 1]=0.12344 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 11 :P[ 0]=0.14242 P[ 1]=0.12083 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 12 :P[ 0]=0.14020 P[ 1]=0.12083 P[ 2]=0.09803 P[ 3]=0.09803
Iter   : 13 :P[ 0]=0.14020 P[ 1]=0.12083 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 14 :P[ 0]=0.14020 P[ 1]=0.12002 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 15 :P[ 0]=0.13952 P[ 1]=0.12002 P[ 2]=0.09709 P[ 3]=0.09709
Iter   : 16 :P[ 0]=0.13952 P[ 1]=0.12002 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 17 :P[ 0]=0.13952 P[ 1]=0.11978 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 18 :P[ 0]=0.13931 P[ 1]=0.11978 P[ 2]=0.09680 P[ 3]=0.09680
Iter   : 19 :P[ 0]=0.13931 P[ 1]=0.11978 P[ 2]=0.09671 P[ 3]=0.09671
```