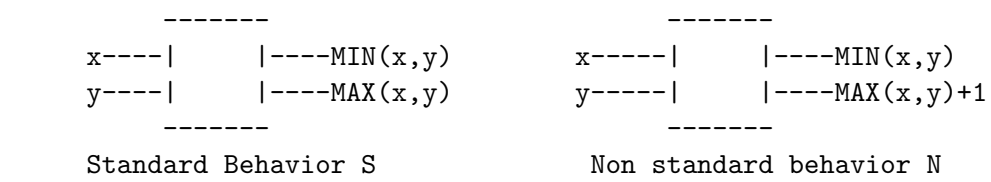
CS 667 : Homework 4(Due: Nov 7, 2005)

Problem 1. (50 points)

In a traditional comparison network every comparator behaves the standard way (let us call it S) that is outlined below. In addition to the standard behavior each comparator may behave as in the non-standard way N also outlined below. For every comparison network C we can flip a switch to decide whether every one of the comparators of C behaves the standard way or the non-standard way.



You are given a comparison network C enclosed in a black box and the only elements of the comparison network that are accessible by you are its input and output lines (i.e. you can set the input lines to specific inputs and read the outputs on the output lines but you can not see what the comparison network looks like) and a switch which you can set it to position S (standard behavior) or N (non-standard behavior).

Show how you can find the size (in terms of number of comparators) of network C even if you are not allowed to peek through the black box.

Problem 2. (50 points)

You are given a real number x, a positive integer n and 2 processors. Design a parallel algorithm that computes x^n in $\lg n + O(1)$ parallel multiplication steps. You are only allowed to use the floating-point/integer operations +, - , * , and the integer operations SHR, LSB.

SHR(x) shifts x right one bit position and inserts a zero into the leftmost bit position. LSB(x) returns the least significant bit of x. For example if $x = 1111$, then after $SHR(x)$, x becomes 0111. If $x = 1110$, $LSB(x)$ is 0.

Problem 3. (50 points)

Let $S = \langle x_1, x_2, \ldots, x_n \rangle$ be a sequence of n distinct keys. The rank of x_1 in the sequence S or $r(x_1, S)$ is the number of keys less than x_1 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?

Can you repeat the two questions above for an EREW PRAM? How would the answers change? Explain.

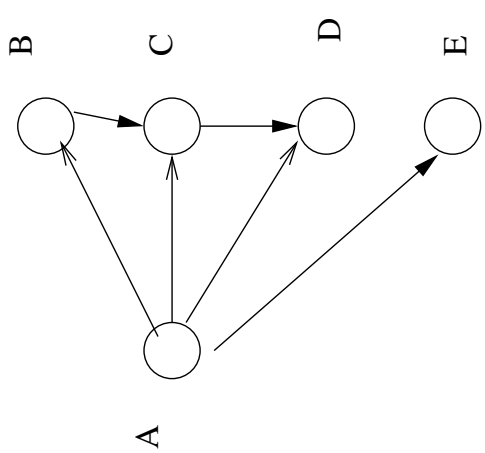


Figure 1.

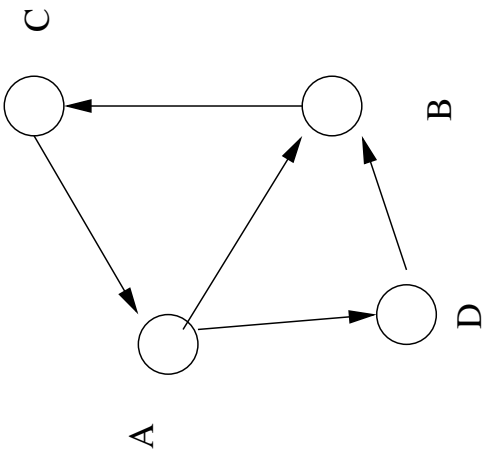


Figure 2.

Problem 4. (50 POINTS)

Find the page rank of the graphs of Figures 1 and 2. Use the algorithm of page 34 of the orginal set of notes for Subject 7 (or page 14 of the new set of notes of Subject 8). 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^{-4}. Use the synchronous version of page 34/14.

Problem 5. (50 POINTS)

Find the hub/authority rank of the graphs of Figures 1 and 2. Initial values are $1/N$. Repeat as many times as long as the error is less than 10^{-4}. Use also the synchronous version of the algorithm. This is almost the algorithm of Page 27 of the old notes of Subject 7 or Page 4 of Subject 8 (`HubAuthority-Rank` with line 4 there modified so that the initial value are $1/N$ rather than $1/N$.

Problem 6. (BONUS 50 POINTS)

Suppose you have a CRCW PRAM and n keys. Find the MIN in $O(1)$ time with $O(n^{1+1/5})$ processors. How does $O(1)$ depend on $1/5$?

Can you do so using $O(n^{1+\epsilon})$ processors for ϵ arbitrarily small and positive constant? How does $O(1)$ depend on ϵ then?

Problem P1. (120 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.
```

Problem P2. (120 points)

Implement Kleinberg's and the PageRank algorithm. The inputs will be graphs represented through an adjacency list.

The command-line interface would be as follows.

```
% ./rank  Algorithm InitialValues Iterations Synchronous InputFile
% java rank Algorithm InitialValues Iterations Synchronous InputFile Sy
```

The command-line parameter `Algorithm` takes one of two values: `0` if Kleinberg's algorithm is used and `1` if the Google PageRank algorithm is used. The second parameter `InitialValues` 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 any other numeric integer value than 0,1,2 then the ranks are initialized as `InitialValues` divided by 100. Thus an `InitialValues` equal to 50, initializes all ranks to $50/100 = 0.5$. Parameter `Iterations` runs the algorithms for that number of iterations. If parameter `Synchronous` is 1 the implementation is synchronous, otherwise it is asynchronous (read the notes and problems 4 and 5 above.) 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.

```
5 6
1 4
1 5
2 4
3 4
3 5
4 1
```

Note 1. For Kleinberg's algorithm you need to implement scaling after all hub and authorities values are updated.