# CS 667 : Homework 4(Due: Nov 18, 2009)

**Problem 1.** (50 points)

Give an efficient algorithm for Parallel_Sum if we have $p$ processors, where $p < n$ ? (You may assume that $n$ is a multiple of $p$, and $n$ is sufficiently large than $p$ say $n/\lg^2 n > p$).

**Problem 2.** (50 points)

(a) Computer $F_n$, the nth Fibonacci number, given an integer $n$ as input. Show how to solve this problem in time $O(\lg n)$ on an EREW PRAM with $n$ processors. Assume that one word of memory is long enough to hold $F_n$. All arithmetic operations (add, subtract, multiply) take constant time. Recall that $F_n$ is defined by the following recurrence: $F_0 = 0, F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.

(b) Give an EREW PRAM algorithm that merges two sorted arrays of size $n/2$ with $P = n/2$ in $O(\lg n)$ time. You may assume that $n$ is a power of two, and you may of course reuse prior results.

**Problem 3.** (50 points)

Can you find the MAX of $n$ keys with $n^{7/6}$ processors in $\Theta(1)$ time? Explain.

**Problem 4.** (50 points)

A search engine retrieves the following three doclists for terms $A$, $B$, $C$. All three are sorted based on docID. Let the sizes of $A$, $B$ and $C$ be $\lg n$, $\sqrt{n}$ and $n$ respectively. $A$, $B$ and $C$ are given in the form of arrays. The output for the problems listed below should be in the same form; it does not need to be sorted however.

(i) We want to find all docs that contain all of $A$, $B$ and $C$.

(ii) We want to find all docs that contain exactly two of $A$, $B$, and $C$.

(iii) We want to find all unique docs that contain $A$ or $B$ or $C$ (i.e. no duplicated docIDs in the output).

Give efficient algorithms for solving each one of these problems. Analyze their worst-case running time. Make them as space efficient as possible. Justify your answers.

**Problem 5.** (50 points)

In this problem we show the effects of parallelizing a "slow" sequential sorting algorithm. The net effect is to obtain a superlinear speedup of more than $p$. The input to the problem is $n$ keys $X = \langle x_0, \ldots, x_{n-1} \rangle$, where $n$ is a multiple of $p$.

**Algorithm** Alg1. Let Alg1 be a sequential $O(n^2)$ running time sorting algorithm of the selection-sort/bubble-sort type. Assume for the remainder that its running time is $T_1(n) = n^2$. This is the baseline algorithm used for speedup purposes.

**Algorithm** Alg2. Alg2 is the following modification of Alg1. It is still a sequential algorithm that will be used to derive a parallel version of Alg1 in the form of Alg3 below.

```
 Alg2 (<x0..xn-1>)
1. Split  <x0..xn-1> into p equally sized groups, each one consisting of n/p keys.
2. Call Alg1 on each one of the p subsequences.
3. Merge the p sorted subsequences of step 2.
```

Show how to do step 3 in $O(n \lg p)$. Then analyze the remaining steps of the algorithm and derive an expression of its worst-case running time in term of $n$ and $p$. For what values of $p$ is the running time $o(n^2)$? Explain. (A little-oh was used, not a Big-oh!)

**Algorithm** Alg3. Alg3 is the following EREW PRAM parallelization of Alg2.

```
 Alg3 (<x0..xn-1>)
1. Processor $i$ is assigned the $i$-th subgroup of consecutive keys <x_in/p .. x_in/p+n/p-1>
   each one consisting of n/p keys.
2. Processor i sorts the i-th group using Alg1 in parallel with all other procs.
3. The  p sorted subsequences  are then merge by processor 0.
```

What is the parallel running time of Alg3? What is its processor count $P$? Explain. For what values of $p$ is the speedup strong superlinear (i.e. $\omega(p)$)? ($\omega$ is not $\Omega$). For which of the following cases is the speedup strong superlinear? (a) $P = \lg n$, (b) $P = \sqrt{n}/\sqrt{\lg n}$, (c) $P = \sqrt{n}$, (d) $P = n/\sqrt{\lg n}$, (e) $P = n$. Explain. How could you characterize (e)? Explain.

**Problem 6.** (80 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 inputs that cause 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 shown 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 7.** (100 POINTS)

Implement the algorithm for the perfect power problem (see Solutions of HW 3/HW 1) in C or C++ or Java.However $n$ can be arbitrarily long (eg. 1024 or 8192 bits). You are allowed to use libraries for arbitrary precision arithmetic as long as (a) they are for free, (b) they are provided in source with the submission, (c) they are installable on AFS (or a Linux Fedora box) at submission time without root privileges. Alternatively, you can implement your own functions for auxiliary operations (eg. arbitrarily long multiplication, exponentiation, etc) or use code from previous homeworks.

I expect as an answer a .tar file sent by email. The .tar will be untarred on a linux workstation and compiled through gcc/g++ or Java. I expect a make install command to compile everything and create an executable file named powertest. powertest will accept as input a file containing $n$ in decimal notation.

Thus if file myfile contains a base-10 integer such as

```
174876867127339284136440637508808648263163265310828908397980471313393
069205071211532685321082692418806710976594639488488379029666130391193
618016247261519151259426686495089936654199866234074092563205917972254
659017817681278771889039846952176691716094827650529253899186783739962
033392687589772827433853061202898692131128514468704543515182864006633
048628011771741733847807753896995603322911363896704685889407210067781
360764270221367332863073641523908250262133391534069401325295056422888
77265570344122667987101745048846965145б
```

then the following command should return

```
% powertest myfile
  x= 123456 y=101
  It's a power!
```

within a reasonable amount of time (eg. under 60 seconds for integers as long as 8192 bits, i.e with up to 2000-3000 digits). `myfile` is a string corresponding to an arbitrarily-named file (in this instance `myfile`).

Note that even if the library you are using has a built-in function to test whether an integer is the integer power of an integer, you are not allowed to use it. You need to build YOUR OWN IMPLEMENTATION from scratch. If you don't you should not expect any credit.