Lindsey Bieda and Joe Frambach
Reduction and Parallel Problems
11.14.2011

15. Give an algorithm for the minimum edit distance problem that runs in poly-log time on a CREW PRAM with with a polynomial number of processors. Here poly-log means O($log^k n$) where n is the input size, and k is some constant independent of the input size.
Recall that the input to this problem is a pair of strings $A = a_1 \dots a_m$ and $B = b_1 \dots b_n$. The goal is to convert A into B as cheaply as possible. The rules are as follows. For a cost of 3 you can delete any letter. For a cost of 4 you can insert a letter in any position. For a cost of 5 you can replace any letter by any other letter.

17. Design a parallel algorithm that finds the maximum number in a sequence $x_1, \ldots, x_n$ of (not necessarily distinct) integers. Your algorithm should run in time $O(\log \log n)$ on a CRCW PRAM with n processors. HINT: Recall that you can find the maximum of k numbers in $O(1)$ time with $\Omega(k^2)$ processors. Try divide and conquer into $\sqrt{n}$ subproblems.

18. Design a parallel algorithm that finds the maximum number in a sequence $x_1, \ldots, x_n$ of (not necessarily distinct) integers in the range 1 to n. Your algorithm should run in constant time on a CRCW Priority PRAM with n processors. Note that it is important here that the x_i's have restricted range. In a CRCW priority PRAM, each processor has a unique positive integer identifier, and in the case of write conflicts, the value written is the value that the processor with the lowest identifier is trying to write.

ParFor $i = 1$ to n:
    Existance$[x_i] = 1$

ParFor $i = 1$ to n:
    Write $n - i$ to Output if Existance$[x_i]$

First we create an existance array to determine which numbers in the range of $1 \ldots n$ exist. Then we make use of the Priority PRAM and write to the output the highest number that exists with the processor that has the lowest identifier (that is writing).

20. Show that if there is an an algorithm for a particular problem that runs in time T (n, p) on a p processor CRCW machine, then there is an algorithm for this problem that runs in time T (n, p) log p on a p processor EREW machine,
Hint: So you need to come up with an algorithm for the following problems:

(a) At some particular time, some subcollection P of processors want to read from a particular location x. The processors in P do not know the identity, or even the number of processors in P. Yet the processors in P need to work together to learn the value stored in location x in time $O(\log p)$.

(b) At some particular time, some subcollection P of processors want to write a common value to a particular location x. The processors in P do not know the identity, or even the number of processors in P. Yet the processor in P need to work together to write the value to in location x in time $O(\log p)$.