Lindsey Bieda and Joe Frambach
Parallel Problems
11.09.2011

9. The input to this problem is a character string C of n letters. The problem is to find the largest k such that

$$C[1]C[2]\dots C[k] = C[n-k+1]\dots C[n-1]C[n]$$

That is, k is the length of the longest prefix that is also a suffix. Give a EREW parallel algorithm that runs in poly-logarithmic time with a polynomial number of processors.

This solution relies heavily on the solution to problem 3a: using n processors to create n copies of an input k in O($log\ n$) time. I will just use the result as if it were a simple operation.

The input is: $c_1, c_2, c_3, \dots, c_{n-2}, c_{n-1}, c_n$.
We can use n^2 processors to create n copies of the input in O($log\ n$) time:

$$\begin{array}{ccccccc}
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n \\
 & & & \vdots & & & \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n
\end{array}$$

Then, we do that again, except each processor has an "offset" 1 to n for the write location. The figure below should be clear.

$$\begin{array}{cccccccccc}
 & c_1 & c_2 & \dots & c_{n-3} & c_{n-2} & c_{n-1} & c_n & & \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n & & & \\
 & & & & & & & & & \\
 & & c_1 & \dots & c_{n-4} & c_{n-3} & c_{n-2} & c_{n-1} & c_n & \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n & & & \\
 & & & & & & & & & \\
 & & & \vdots & & & & & & \\
 & & & & & & & & & \\
 & & & & & c_1 & c_2 & c_3 & \dots & c_{n-1} & c_n \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n & & & & \\
 & & & & & & & & & \\
 & & & & & & c_1 & c_2 & \dots & c_{n-1} & c_n \\
c_1 & c_2 & c_3 & \dots & c_{n-2} & c_{n-1} & c_n & & & &
\end{array}$$

Next, we use n processors on each "pairing" above, for a total of n^2 processors. Each processor writes a 1 if the aligning characters are equal, otherwise write 0.

Next, we have n AND-problems. Each pairing contains 1 to n ones or zeroes. If a prefix equals a suffix, then all comparisons yielded 1. The AND-problem for n inputs is solved with n processors in $log\ n$ time (Problem 1).
Now, we have to find the MAX of the alignment-lengths of the pairings which yielded 1 for the comparison. This is again done in $log\ n$ time with n processors using the same method as every other problem we've seen so far.

10. The input to this problem is a character string C of n letters. The problem is to find the largest k such that

$$C[1]C[2]\dots C[k] = C[n-k+1]\dots C[n-1]C[n]$$

That is, k is the length of the longest prefix that is also a suffix. Give a CRCW parallel algorithm that runs in constant time with a polynomial number of processors.

We will need $n/2$ processors in order to determine if there is a prefix and a suffix at most of length $n/2$. This is the maximum length to consider since any prefix longer than $n/2$ has less than $n/2$ as the remaining possible suffix and therefore the prefix and suffix cannot be equivalent. Each of these $n/2$ processors check for a prefix and suffix that is of length $\frac{n}{2}, \frac{n}{2}-1, \frac{n}{2}-2, \dots, 1$. Since we write 0 initially to the solution location there is no need to verify. For each of these processors they check if the string at the beginning and the end of the given length is equal and then write to the memory location if and only if the k value stored there is less than the value being written.

In order to verify the equality of these two substrings we need k processors each required to look in one index in the two string arrays. The equality bit starts at 1 and if at the same index two letters differ a 0 is written.

11. Design a parallel algorithm for adding two n-bit integers. You algorithm should run in O($log\ n$) time on a CREW PRAM with n processors.
NOTE: If your algorithm is EREW, you might want to rethink since I don't know how to do this easily with out CR.
HINT: Use divide and conquer and generalize the induction hypothesis.