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

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

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

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

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

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

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.

Consider the following setup from the previous problem:

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

Since we have a CRCW machine, the instances of c_i can be considered references rather than copies.
The rest of the operations are exactly the same as the previous problem, with modifications for CRCW.
The AND-problem is constant-time with n processors per "pairing".
The MAX operation is the same as the MIN operation, which was done in class and is available in the course notes. It requires n^2 processors, comparing each pairing of numbers, returning 1 if the first is greater, 0 otherwise. The AND operation is run on each row. The row containing no zeroes is the MAX.

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.