Homework 29

Brian Knotten, Brett Schreiber, Brian Falkenstein

November 4, 2018

9

10

The general outline for this algorithm is as follows:

At the first time step, use $n-1$ processors to write the numbers $1, 2, 3, 4, ...n-1$ in memory location M. These represent values for k such that there exists a matching prefix and suffix of size k.

At the second time step, use n^2 processors to compare the prefixes and suffixes of all lengths for k from 1 to $n-1$. If the prefix and suffix aren't equal to each other, then "zero out" the location in memory. For example, if the first 3 characters do not match the last 3 characters, then the numbers in M become $1, 2, 0, 4, ...n-1$.

Finally, at the third time step, use n^2 processors to find the maximum number left in M. This will return the maximum valid k, and it can be done in constant time as discussed in class.

Below is an algorithm for one of the $(n-1)^2$ processors, i, j. (Without loss of generality, the processors are labelled with two numbers, $i \in [1...n-1]$ and $j \in [1...n-1]$ for easier usage).

Algorithm 1 CRCW Common $O(1)$ algorithm

Require: A string C of size n, a processor $p_{i,j}$, a memory location M of size $n-1$, a memory location T of size $(n-1)^2$, and a memory location And of size $n-1$.

```
M[i] ← i                          ▷ First, copy the numbers 1, 2, 3...n − 1 into M.
if C[j] ≠ C[n − i + j] then
    M[i] ← 0                      ▷ If any of the pairs of characters don't match, then that k isn't viable.
end if
if M[i] ≥ M[j] then               ▷ Perform all possible pairwise comparisons of M.
    T[i, j] ← 1                   ▷ Record which indices are greater.
else
    T[i, j] ← 0
end if
And[i] ← 1                        ▷ Perform an EREW AND operation to find a row in T of all 1s.
if T[i, j] = 0 then
    And[i] ← 0
end if
if And[i] = 1 then
    Output i                      ▷ That row is the maximum k. A maximum always exists, so this will always output.
end if
```

11