# Homework 29

Brian Knotten, Brett Schreiber, Brian Falkenstein

November 4, 2018

## 9

The general outline of the algorithm is as follows:

Due to the EREW restriction, the algorithm must first make copies of the input string $C$. The algorithm uses $n^2$ processors to make $n^2$ copies of the string in $log(n)$ time. It does this by using $x * n$ processors to make $x$ copies of the input string, where each processor reads a character from the string and writes it to an array. The array is of size $n \times n^2$, each row being one copy of $C$. The array will also be referred to as $C$, as it can be thought of as adding another dimension to the input string $C$, where each added row is a copy of $C$. Note that when $x > n$, we cannot make $xn$ copies in one step, as $x * n > n^2$. However, the additional $((x * n) - n^2)$ copies can be done in constant time. This makes the whole initial copying take $log(n)$.

Next, $k$ processors are used to write an answer array, we'll call it $M$, where the $i'th$ index of $M$ is $i$ if there exists a prefix and suffix of length $i$, or 0 otherwise.

Then, $n^2$ processors are used to check all possible prefix/suffixes of lengths 1 to $k$. Each processor is identified by 2 numbers $i, j$. If its found that there is no prefix suffix of length $k = i$, write a zero to $M[i]$.

Next, the max of the $M$ array must be found. Because $M[i] = i$ iff a prefix and suffix exist of length $i$, and 0 otherwise, if we find the max of $M$, we will find the max length of prefix/suffix, and thus $k$.

Below is the algorithm for a single processor $i, j$:

## 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).

This algorithm runs in $O(1)$ time, since each processor only performs a constant number of operations, as described above.

## 11

**Algorithm 1** EREW $O(log(n))$ algorithm

**Require:** A string $C$ of size $n$ to be expanded to an $n^2 \times n$ array of copies, a processor $p_{i,j}$, a memory location $M$ of size $n-1$

**if** $j == 1$ **then**
  $M[i] \leftarrow i$ ▷ Have $n$ processors write $1...k$ to $M$
**end if**
$number\_of\_copies \leftarrow 1$ ▷ Variable to store the current number of copies made
**while** $number\_of\_copies < n^2$ **do**
  **if** $j < number\_of\_copies$ **then** ▷ Only use the processors needed to make $c$ copies
    $C[i][j + number\_of\_copies] \leftarrow C[i][j]$ ▷ Copy current character to new copy location
  **end if**
  **if** $number\_of\_copies > n$ **then** ▷ Can't double the number of copies, can only make $n$ more
    $number\_of\_copies \leftarrow number\_of\_copies + n$
  **else**
    $number\_of\_copies \leftarrow number\_of\_copies * 2$
  **end if**
**end while**
**if** $C[j][2i + j] \neq C[n - i + j]$ **then**
  $M[i] \leftarrow 0$ ▷ If any of the pairs of characters don't match, then that $k$ isn't viable.
**end if**
$y \leftarrow \lfloor n/2 \rfloor$ ▷ Now get the max of the $M$ array
**while** $i < y$ **do**
  $M[i] \leftarrow MAX(M[i], M[i + y])$ ▷ A processor can take the max of 2 values in constant time
  $y \leftarrow \lfloor y/2 \rfloor$
**end while**
**if** $i == 0$ and $j == 0$ **then** ▷ Have one processor output the solution
  output $M[0]$
**end if**

**Algorithm 2** CRCW Common $O(1)$ algorithm

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

$M[i] \leftarrow i$ ▷ First, copy the numbers $1, 2, 3...n-1$ into $M$.
**if** $C[j] \neq C[n - i + j]$ **then**
  $M[i] \leftarrow 0$ ▷ If any of the pairs of characters don't match, then that $k$ isn't viable.
**end if**
$And[i] \leftarrow 1$ ▷ Perform an EREW AND operation to find a row in $T$ of all 1s.
**if** $M[i] < M[j]$ **then** ▷ Perform all possible pairwise comparisons of $M$ using $n^2$ processors.
  $And[i] \leftarrow 0$ ▷ If $M[i]$ is less than any $M[j]$, then $M[i]$ cannot be the maximum $k$.
**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**