Homework 30

Brian Knotten, Brett Schreiber, Brian Falkenstein

November 6, 2018

12

In the CREW algorithm, the concurrent read step is $D[i, j] \leftarrow min(D[i, j], D[i, m] + D[m, j])$, since processor $i = 5$ is reading $D[i = 5, m = 4]$ at the same time processor $j = 4$ is reading $D[m = 5, j = 4]$ (These represent the same memory location). To make the algorithm EREW, make n copies of D into a 3D array D' so that each of the n^3 processors $p_{i,j,k}$ is reading from a different cell in D'.

n^3 processors can make n copies of D (which is size n^2) into a 3D array D' in $\log^2 n$ time.

With n^3 processors, the concurrent read step of the CREW algorithm becomes an exclusive read step: $D[i, j, k] \leftarrow min(D[i, j, k], D[i, m, k] + D[m, j, k])$. But now each $D[i, j, k]$ must be synchronized over all k, since any one of the ks could have changed the minimum.

```
while c <= 1:
if k<c:
D'[i, j, k+c] = D'[i, j, k]
end
b = floor(c/2)
while x < b do:
D'[i, j, i] = min(D'[i, j, i], D'[i+b, k+b, i] + D'[k+b, j, i])
end
if i == 0 & j == 0 & k == 0
return D'[0, 0, 0]
end
end
```

13

The algorithm can be modified as follows to return the actual paths. First, when setting up the 2D array D that contains information for all pairs $(v_i, v_j) \in G$, instead of storing the distance between v_i and v_j store a pair: the first element is a list of vertices representing a path from v_i to v_j, and the second element is the distance between v_i and v_j.

When setting up D initially, $D[i, j]$ is initialized as follows: if $i = j$, then $D[i, j] \leftarrow ([v_i], 0)$. Else if there is an edge between v_i and v_j, then $D[i, j] \leftarrow ([v_i, v_j], edge_weight(v_i, v_j))$. Otherwise, $D[i, j] \leftarrow ([], \infty)$.

When $D[i, j]$ is updated to be $D[i, m] + D[m, j]$, then $D[i, j] \leftarrow (fst(D[i, m]) + +fst(D[m, j]), snd(D[i, m]) + snd(D[m, j]))$, where fst returns the first element of a tuple, snd returns the second element of a tuple, and $a + +b$ represents list a concatenated with list b.

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^2

$M[i][j] \leftarrow i$ ▷ Use n^2 processors to write $1...k$ into $M[i]$ for all i.
$number\_of\_copies \leftarrow 1$ ▷ This is a 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**
    $number\_of\_copies \leftarrow number\_of\_copies + n$ ▷ All n^2 processors can copy at most n strings eath of length n.
  else
    $number\_of\_copies \leftarrow number\_of\_copies * 2$ ▷ Otherwise, with less than n^2 processors, the number of copies can be doubled.
  end if
end while
if $C[j][2i + j] \neq C[n - i + j][2i + j]$ **then** ▷ Each processor compares two individual characters to see if the prefix is of size k. The extra dimension on C is to ensure that all processors are reading from different places in memory.
  $M[i][j] \leftarrow 0$ ▷ If any of the pairs of characters don't match, then that k isn't viable.
end if
$z \leftarrow \lfloor n/2 \rfloor$ ▷ Flatten the M array so that any zero entry makes $M[i][1] \leftarrow 0$.
while $j < z$ **do**
  $M[i][j] \leftarrow MIN(M[i][j], M[i][j + z])$ ▷ If any entry in the j column is zero, then $M[i][1]$ is 0.
  $z \leftarrow \lfloor z/2 \rfloor$
end while
$y \leftarrow \lfloor n/2 \rfloor$ ▷ Now get the max of the $M[1]$ array
while $i < y$ **do**
  $M[i][1] \leftarrow MAX(M[i][1], M[i + y][1])$ ▷ A processor can take the max of 2 values in constant time. Overwrite the greater number into $M[i]$. After $\log n$ iterations, $M[1][1]$ will contain the maximum k.
  $y \leftarrow \lfloor y/2 \rfloor$
end while
if $i == 1$ and $j == 1$ **then** ▷ Designate the first processor to exclusively read and output the solution.
  Output $M[1][1]$
end if