Homework 27

Robbie McKinstry, Jack McQuown, Cyrus Ramavarapu

04 November 2016

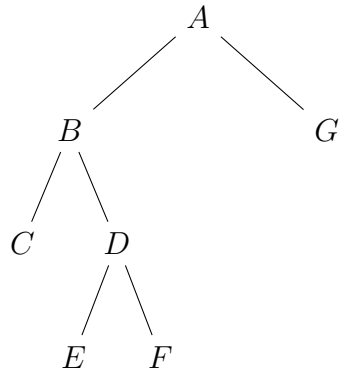
Problem 21:

On an EREW PRAM, the leave nodes of an arbitrary tree with n nodes can be numbered in order in $\mathcal{O}(\log n)$ time using n processors if initially each of the n processors is assigned to a unique node.

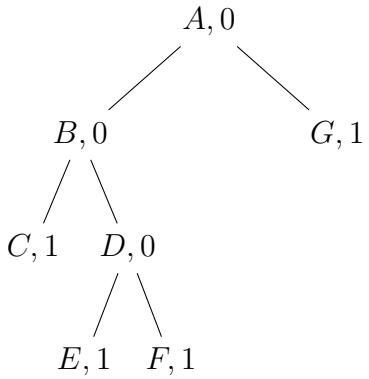
Initially, each processor will have to determine whether or not it is assigned a leaf node. This can be done in parallel in $\mathcal{O}(1)$ time by having each processor check if has any children. If the processor has children, it will hold the value 0. If the processor is assigned to a leaf node it will hold the value 1.

After numbering all the internal nodes as 0 and all the leaf nodes as 1, an *Eulerian Tour* can be performed starting from the root of the tree headed left. Each edge will be marked with a +0 and the result of the tour will be a linked list where each node will either contribute 1 to the sum, if it is a leaf, or a 0 if it is an internal node. The parallel prefix problem can then be solved on this linked list, and the resulting sequence of consecutive numbers will number the leaves from left to right.

To demonstrate this algorithm, consider the following tree:



After the initial step that will number each node as either a 0 or 1 depending if the node is a leaf or an internal node, the tree will look as follows:



The *Eulerian Tour* of the tree will begin at A and then track values as makes progress beginning from the left. This linked list will look like the following.

Using the pointer doubling technique, the parallel prefix problem on the linked list can be solved in $\mathcal{O}\big(\log n\big)$ time. Since all the other steps were doable in $\mathcal{O}\big(1\big)$ time, the overall runtime of the algorithm will be $\mathcal{O}\big(\log n\big)$.

Problem 22:

On an EREW PRAM, the balance factor for each node of an arbitrary tree with n nodes can be done in $\mathcal{O}\big(\log n\big)$ time using n processors if initially each

of the n processors is assigned to a unique node.

Initially, each processor will have to know the height of the node it is pointing to. In order to find the height of each node in the tree we can do an *Euler Tour* of the tree and have each node hold the value of its respective height like we did in class. The process for finding the height of each node is by keeping a sum which increments when going down a level and decrements when going back up a level. The sum is then stored at each node when the node is visited. This *Euler Tour Technique* can be done in $\mathcal{O}(1)$ time, which then leaves the process of finding the balance factor at each node.

To find the balance factor for each node, we can use a simple equation $Balance(x) = Height(x.left) - Height(x.right)$. This Balance(x) calculation can be done in parallel in that we can start from the leaves and work our way up the tree and calculate the balance for every node at a certain level in the tree.

The leaves of the tree will initially have a balance set to 0, which can be done in $\mathcal{O}(1)$ time. We can then progress up the tree and at each level calculate the Balance(x) for each node at that level in parallel. Using this technique we can achieve a runtime of $\mathcal{O}(\lg n)$.