The Euler-tour technique

The problem of computing the depth of each node in an n-node binary tree

Binary tree

- Let T be a binary tree stored in a PRAM
- Each node *i* has fields parent[i], left[i] and right[i], which point to node i's parent, left child and right child respectively
- Let's assume that each node is identified by a non-negative integer
- Also we associate not one but 3 processes with each node; we call these node's A,B and C processors
- Mapping between each node *i* and its 3 processors A,B and C: 3i, 3i+1, 3i+2

Computing depth of each node in an n node tree takes O(n) time on a serial RAM

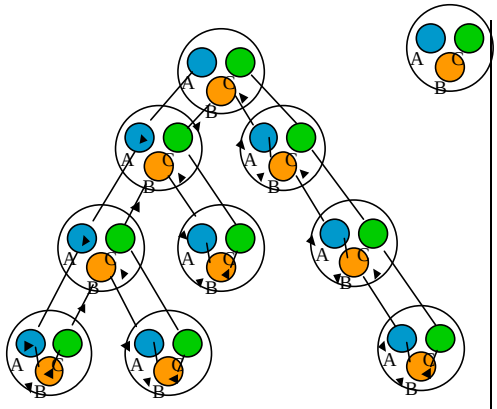
- A simple parallel algorithm to compute depths propagates a "wave" downward from the root of the tree.
  - The wave reaches all nodes at the same depth simultaneously, and thus by incrementing a counter carried along with the wave, we can compute the depth of each node.
- This parallel algorithm works well on a complete binary tree, since it runs in time proportional to the tree's height.
- But the height of the tree could be as large as n-1

Using the Euler-tour technique we can compute node depths in O(log n) time on an EREW PRAM

- An Euler-tour of a graph is a cycle that traverses each edge exactly once, although it may visit a vertex more than ones
  - A connected, directed graph has an Euler tour if and only if for all vertices v, the in-degree of v equals the out degree of v
  - Since each undirected edge (u,v) in an undirected graph maps to two directed edges (u,v) and (v,u) in the directed version, the directed of any connected, undirected graph (and therefore of any undirected tree) has an Euler tour

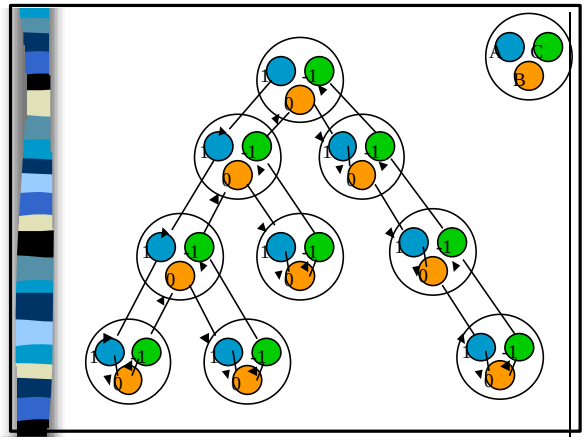
Depth of nodes computation

- First we form an Euler tour of the directed version of T.
- The tour corresponds to walk of the tree with the following structure:
  - A node's A processor points to the A processor of its left child, if it exist, and otherwise to its own B processor
  - A node's B processor points to the A processor of its right child, if it exist, and otherwise to its own C processor
  - A node's C processor points to the B processor of its parent, if it is a left child and to the C processor of its parent if it is a right child. The root's C processor points to NIL.



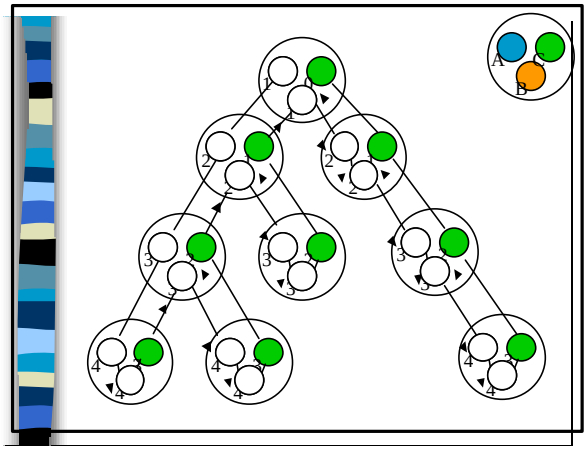
First step

- Thus, the head of the linked list formed by the Euler tour is the root's A processor, and the tail is the root's C processor.
- Given the pointers composing the original tree, an Euler tour can be constructed in O(1) time.
- Once we have linked list representing the Euler tour of T, we place
  - a 1 in each A processor,
  - a 0 in each B processor and
  - a –1 in each C processor



Second step

- We then perform a parallel prefix computation using ordinary addition as the associative operation
- We claim that after performing the parallel prefix computation, the depth of each node resides in the node's C processor. Why?



WHY ???

- The numbers are placed into the A,B and C processors in such a way that the net effect of visiting a subtree is to add 0 to the running sum
- The A processor of each node *i* contributes 1 to running sum
- The B processor of each node *i* contributes 0 because the depth of the node *i*'s left child equals the depth of the node *i*'s right child
- The C processor contributes –1, so the entire visit to the subtree rooted at node *i* has no effect on the running sum.

Conclusion

- The list representing Euler-tour can be computed in O(1) time.
- It has 3n objects, and thus the parallel prefix computation takes only O(log n) time
- Thus the total amount of time to compute all node depths is O(log n).
- Because no concurrent memory accesses are needed, the algorithm is an EREW algorithm.