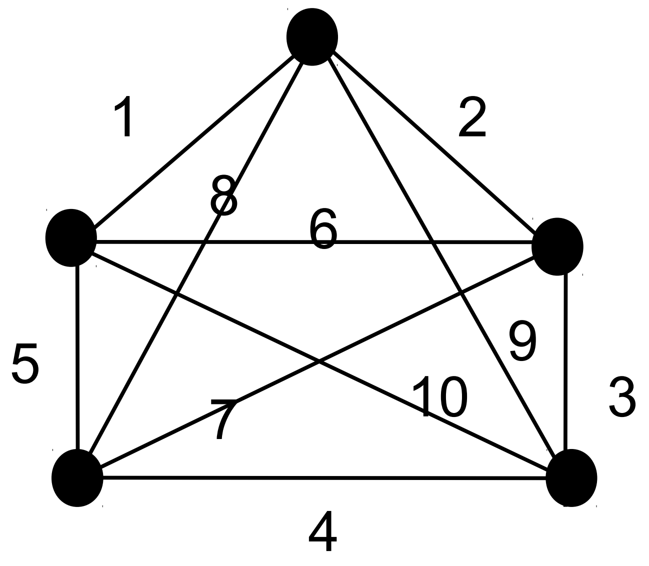
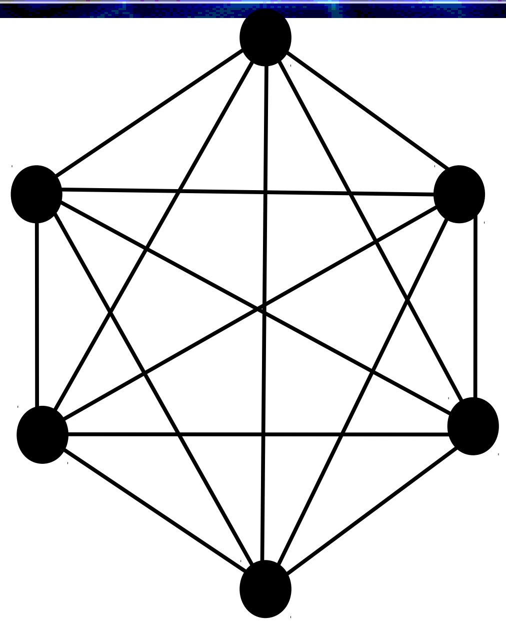
Tree Processing

- Now that we know how to process linked lists, let us consider tree algorithms.
- Problems we will consider:
  - Traversal
  - Expression evaluation
  - Least common ancestor, range minima

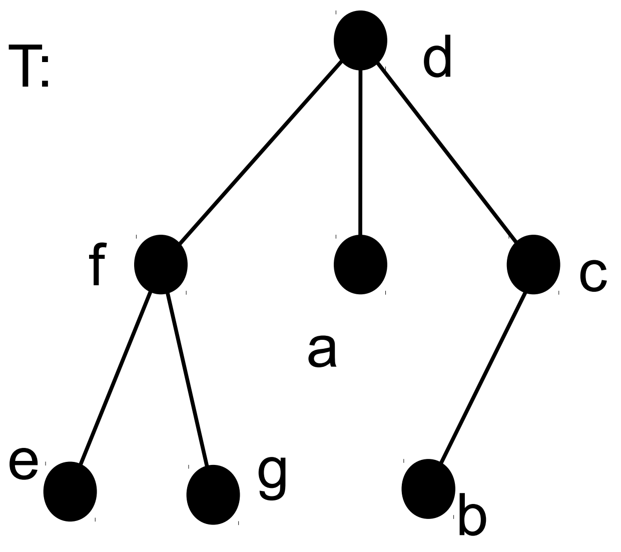
Traversal via Euler Tour

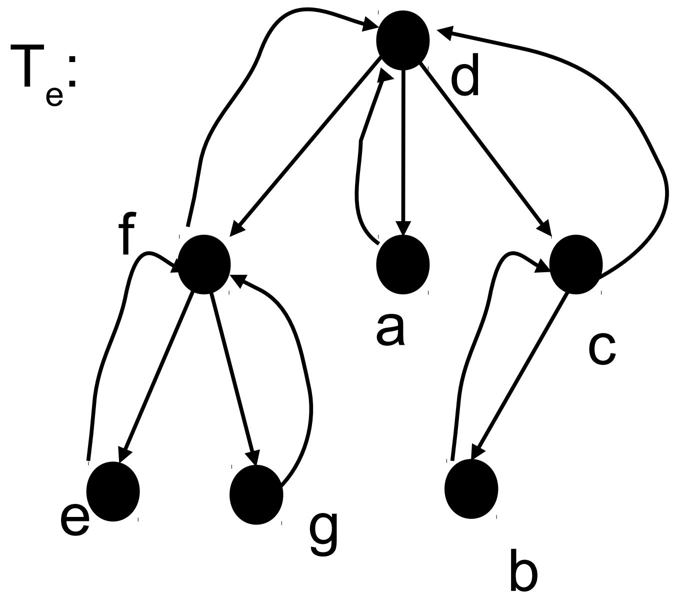




- Given a tree T = (V, E), we use an Euler tour as a primitive for tree traversal algorithms.
- Euler tour: Given a graph, an Euler tour is a cycle that includes every edge exactly once.
- A directed graph G has an Euler tour if and only if for every vertex, its in-degree equals its out-degree.

Euler Tour of a Tree





- For a tree T = (V,E) to define an Euler tour, we make it a directed tree:
  - Define $T_e = (V_e, E_e)$ with $V_e = V$.
  - For each edge uv in E, add two directed edges (u,v) and (v,u) to E_e.
  - T_e has an Euler tour.

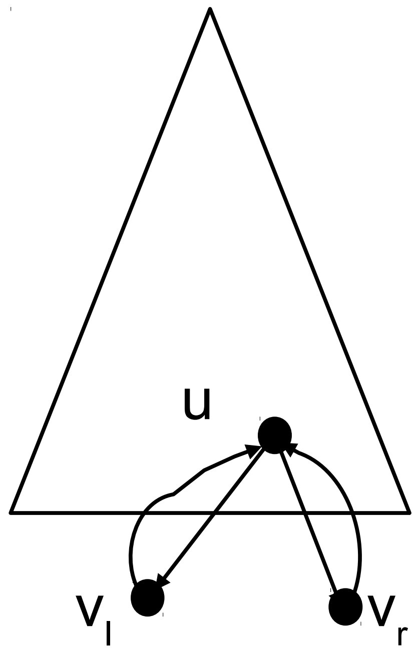
Defining an Euler Tour on a Tree

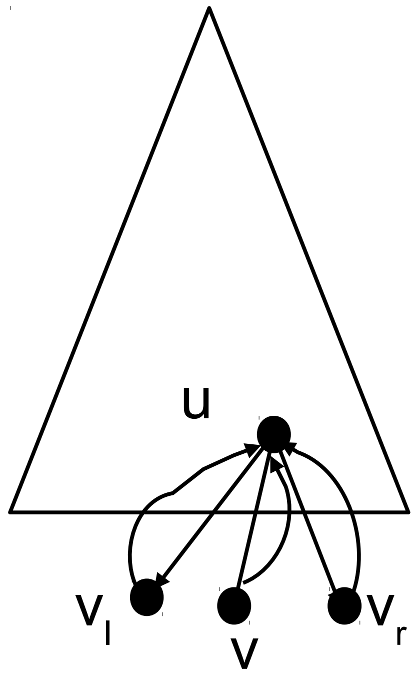
- Just have to define a successor. Here, successor for an edge.
- For a node u in T_e order its neighbors v_1, v_2, ..., v_d.
  - Can be done independently at each node.
  - For e = (v_i, u), set s(e) = e' where e' = (u, v_{i+1}).
    - Compute indices modulo the degree of u.

Euler Tour on a Tree

- Claim: The above definition of s: E → E is a tour.
- Proof: By induction. Let n = 1. Obviously true.
- For n = 2, at most one edge present. The tour is well defined according to s().
- Let the tour be well defined for n = k.
- Step: n = k+1.
  - Every tree has at least one leaf, say v.
  - T' = T \ {v} has an Euler tour defined by s':E(T') → E(T') as |V(T')| = k.
  - We now extend this definition of s' to define a function s: E(T) →E(T).

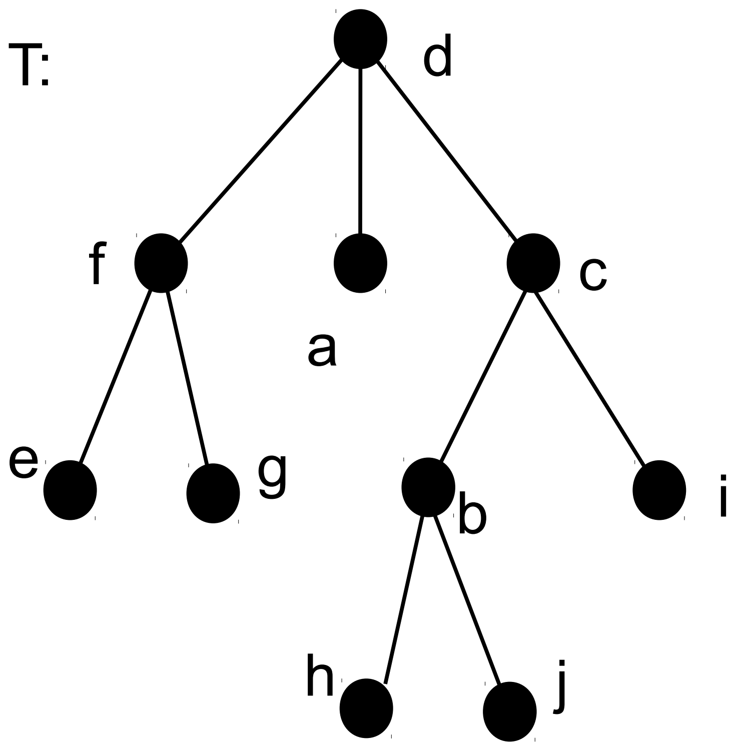
Euler Tour on a Tree





- Let u be the neighbor of v in T.
- Let N(u) = $\{v_0, \ldots, v_{i-1}, v_i=v, v_{i+1}, \ldots v_d\}$.
- Set s(u, v) := (v,u). Set s(v_{i-1},u) := (u,v).
- At all other edges $e \in T$, s(e):=s'(e).

Example



$$\begin{tikzcd}
a \arrow[r] & d & & \\
b \arrow[r] & j \arrow[r] & h \arrow[r] & c \\
c \arrow[r] & i \arrow[r] & d \arrow[r] & b \\
d \arrow[r] & c \arrow[r] & a \arrow[r] & f \\
e \arrow[r] & f & & \\
f \arrow[r] & g \arrow[r] & d & \\
g \arrow[r] & f & & \\
h \arrow[r] & b & & \\
i \arrow[r] & c & & \\
j \arrow[r] & b & &
\end{tikzcd}$$

Example

$$\begin{tikzcd}
a \arrow[r] & d \\
b \arrow[r] & j \arrow[r] & h \arrow[r] & c \\
c \arrow[r] & i \arrow[r] & d \arrow[r] & b \\
d \arrow[r] & c \arrow[r] & a \arrow[r] & f \\
e \arrow[r] & f \\
f \arrow[r] & g \arrow[r] & d \\
g \arrow[r] & f \\
h \arrow[r] & b \\
i \arrow[r] & c \\
j \arrow[r] & b
\end{tikzcd}$$

s(b,c) = (c,i)

s(c,i) = (i,c)

s(i,c) = (c,d)

s(c,d) = (d,a)

s(d,a) = (a,d)

s(a,d) = (d,f)

s(d,f) = (f,g)

s(f,g) = (g,f)

s(g,f) = (f,d)

s(f,d) = (d,c)

s(d,c) = (c,b)

s(c,b) = (b,j)

s(b,j) = (j,b)

s(j,b) = (b,h)

s(b,h) = (h,b)

s(h,b) = (b,c)

s(b,c) = (c,i)

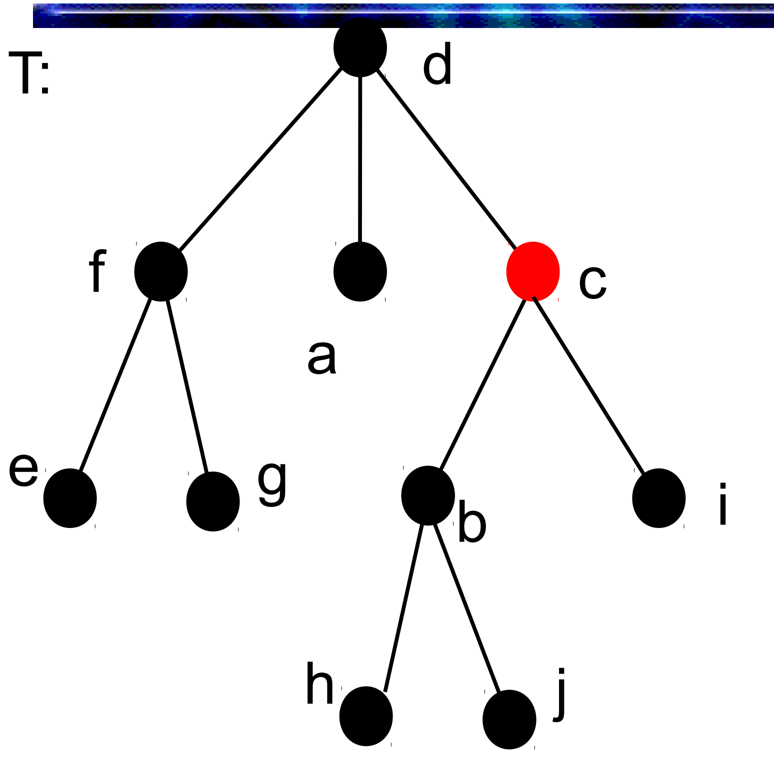
Euler Tour:

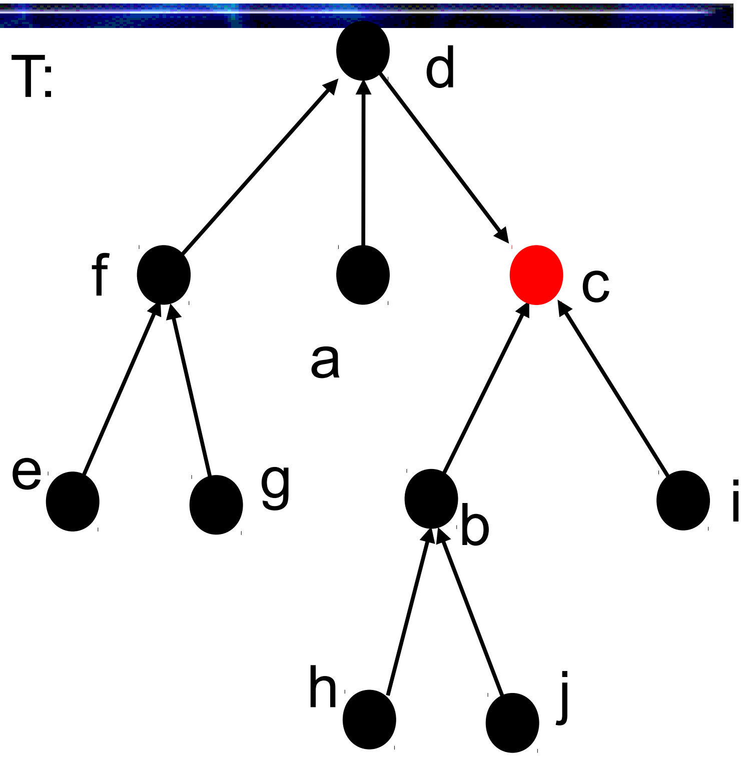
(b,c) → (c,i) → (i,c) → (c,d) → (d,a) → (a,d) → (d,f) → (f,g) → (g,f) →
(f,d) → (d,c) → (c,b) → (b,j) → (j,b) → (b,h) → (h,b) → (b,c)

Applications of Euler Tour to Traversal

- We now show why the Euler tour is an important construct for trees.
- Operations on a tree such as rooting, perorder and postorder traversal can be converted to routines on an Euler tour.

Rooting a Tree

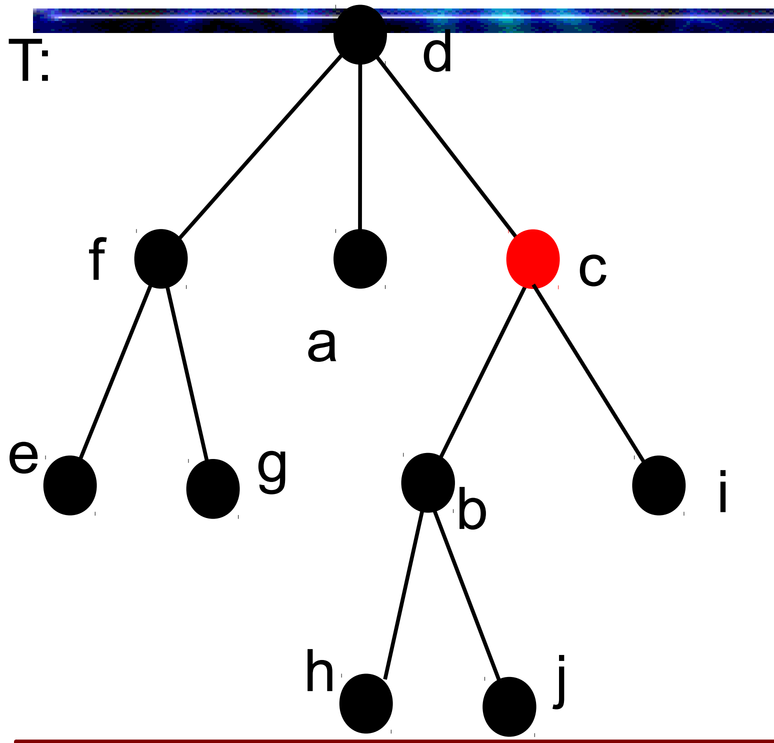


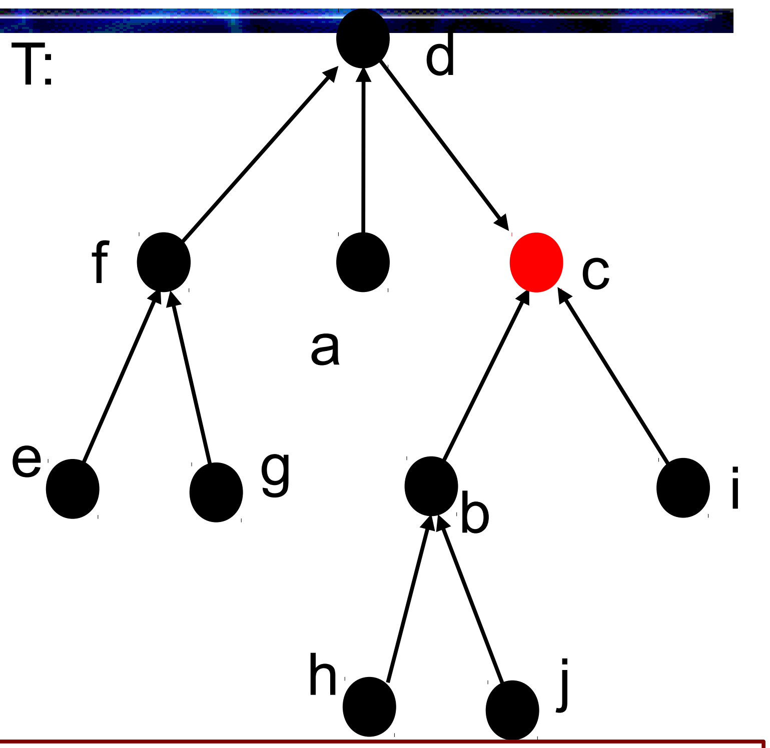


Euler Tour:
(b,c) → (c,i) → (i,c) → (c,d) → (d,a) → (a,d) → (d,f) → (f,g) → (g,f) → (f,d) → (d,c) → (c,b) → (b,j) → (j,b) → (b,h) → (h,b) → (b,c)

- Designate a node in a tree as the root.
- All edges are directed towards the root.

Rooting a Tree



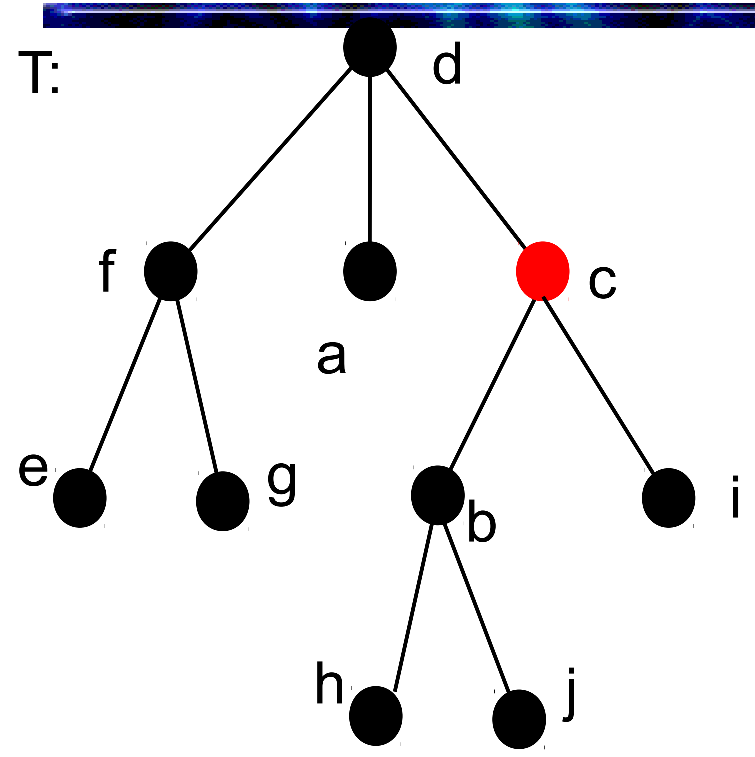


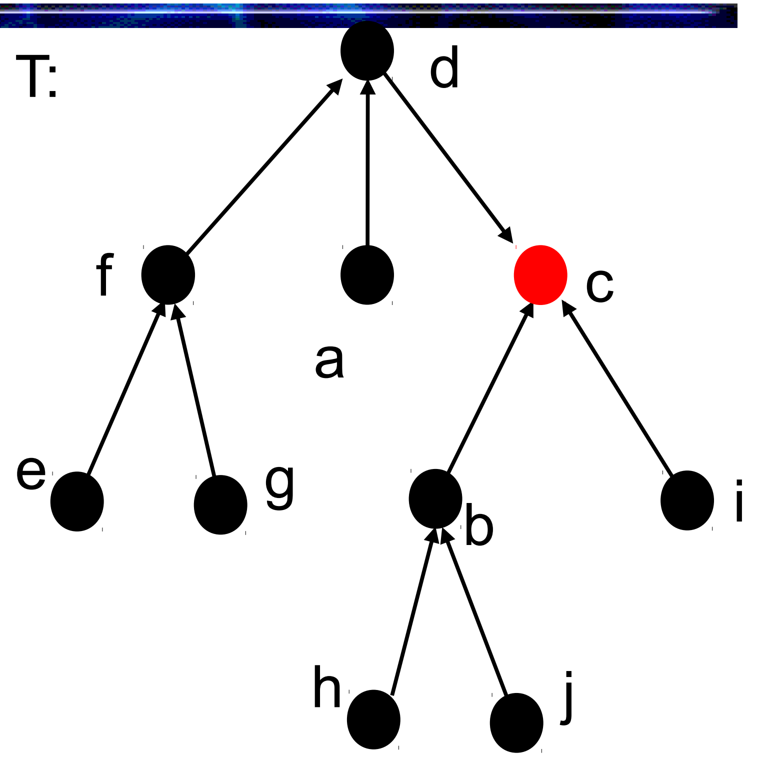
Euler Tour:

(b,c) → (c,i) → (i,c) → (c,d) → (d,a) → (a,d) → (d,f) → (f,g) → (g,f) → (f,d) → (d,c) → (c,b) → (b,j) → (j,b) → (b,h) → (h,b) → (b,c)

- Let (v_1, v_2, ..., v_d) be the neighbors of the root node r. In this case, say (i, d, b)
- Set s(e) = NULL for e = (v_d, r). In this case, s((b,c)).

Rooting a Tree



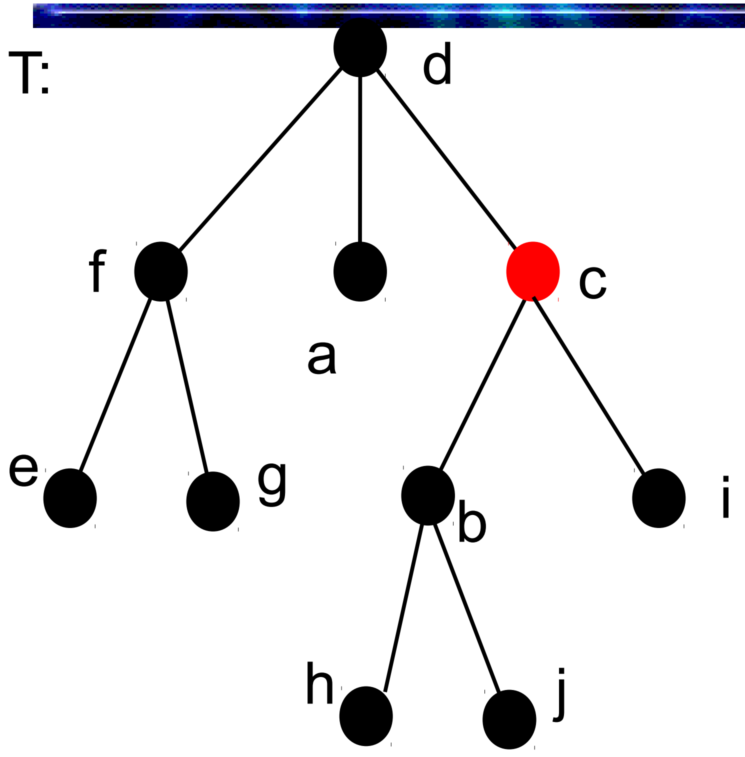


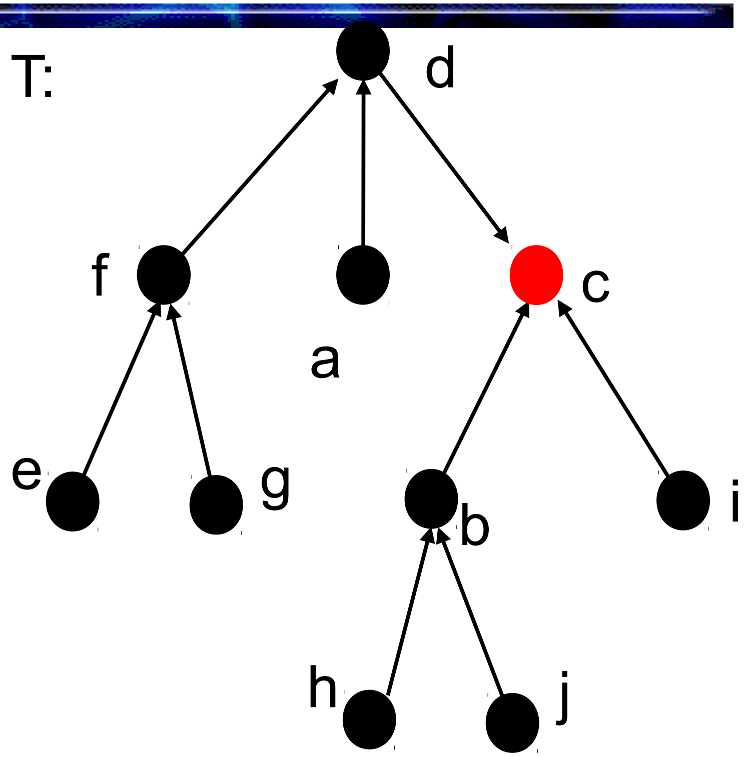
Euler Tour:

(c,i) → (i,c) → (c,d) → (d,a) → (a,d) → (d,f) → (f,g) → (g,f) → (f,d) → (d,c) → (c,b) → (b,j) → (j,b) → (b,h) → (h,b) → (b,c) → NIL

- The edge (r,v_i) appears before (v_i,r).
- So, if uv precedes vu, then set u = p(v). Orient the edge uv from v to u.

Rooting a Tree



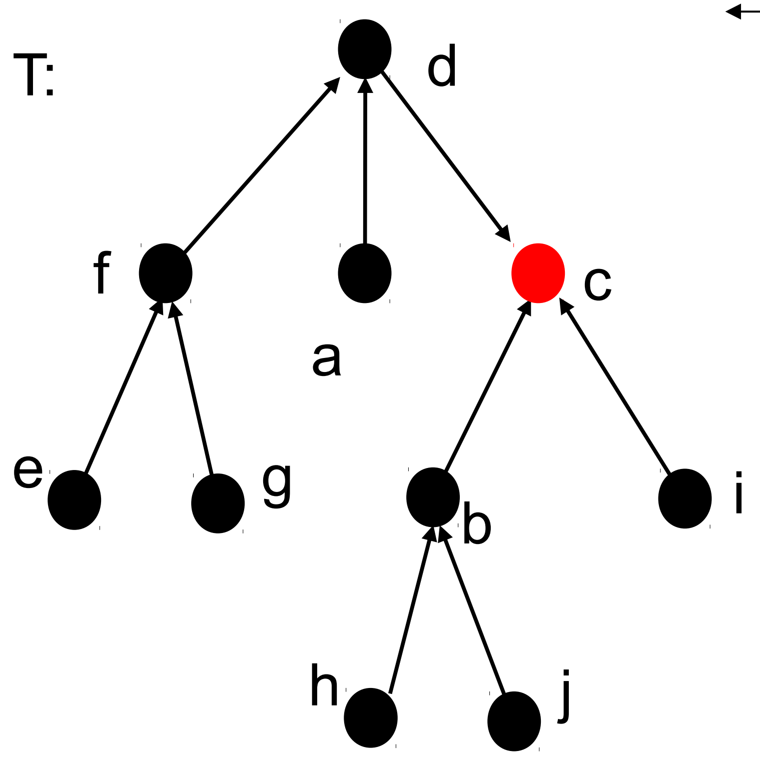


Euler Tour:

(c,i) → (i,c) → (c,d) → (d,a) → (a,d) → (d,f) → (f,g) → (g,f) → (f,d) → (d,c) → (c,b) → (b,j) → (j,b) → (b,h) → (h,b) → (b,c) → NIL

- So, if uv precedes vu, then set u = p(v). Orient the edge uv from v to u.
- To know the above, use list ranking on the Euler path.

Preorder Traversal



ci-ic-cd-da-ad-df-fe-ef-fg-gf-fd-dc-cb-bj-jb-bh-hb-bc

$\longleftrightarrow$ Euler path Entering node d and its subtree

$\longleftrightarrow$ Euler path at node b and its subtree

- Euler tour can be used to get a preoder number for every node.
- Associate meaning to the Euler tour.

Preorder Traversal

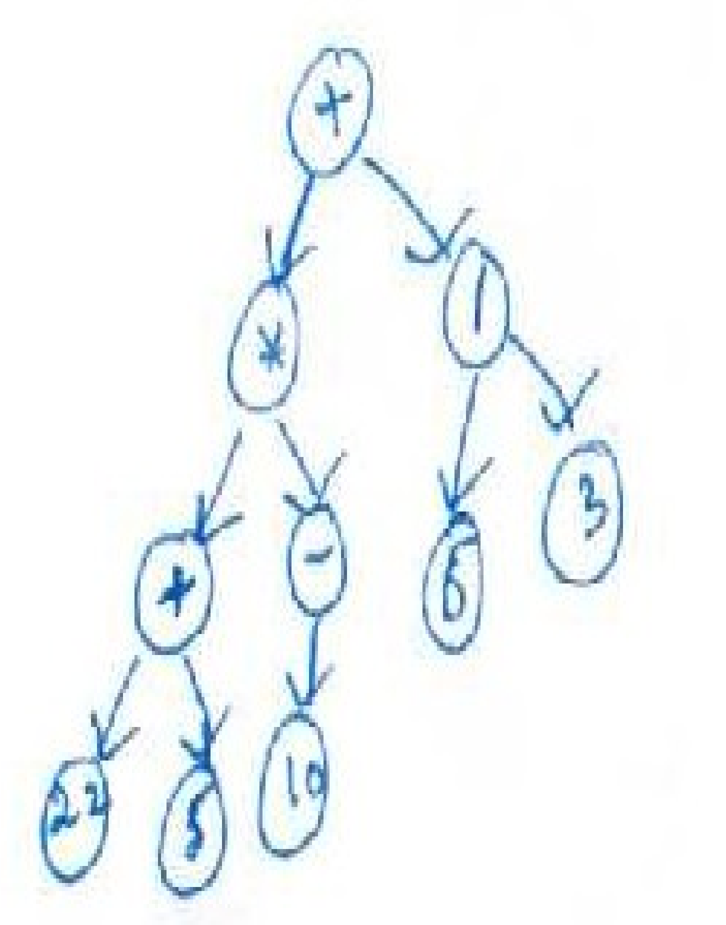
- In preorder traversal, a node is listed before any of the nodes in its subtree.
- In an Euler tour, nodes in a subtree are visited by entering those subtrees, and finally exiting to the parent.
- If we can therefore track the first occurrence of a node in the Euler path, then we can get the preorder traversal of the tree.
    - In the Euler path the first occurrence of a node v is via the edge (p(v),v).

Postorder Traversal

- Similar rules can be designed for also postorder and inorder traversals.
- Inorder for binary trees only makes sense.
- Next we see how to process expression trees.

Expression Trees

- Expression trees are trees with operands at the leaf nodes, and operators at the internal nodes.
- Our interest is to evaluate the result of an expression represented by its expression tree.
- We would limit ourselves to binary operators.
- Can also convert non-binary cases to the case of binary operators.
- However, the expression tree need not be balanced, or height in Θ(log n).



Expression Tree Evaluation

- Cannot directly apply the standard technique of

  All the penultimate level nodes, then the next level etc.

  - Tree may not be balanced, and could have very few nodes at the penultimate level.
  - All leaves first cannot be processed at once
    - At an internal node, both operands may not be evaluated yet.

Two Steps

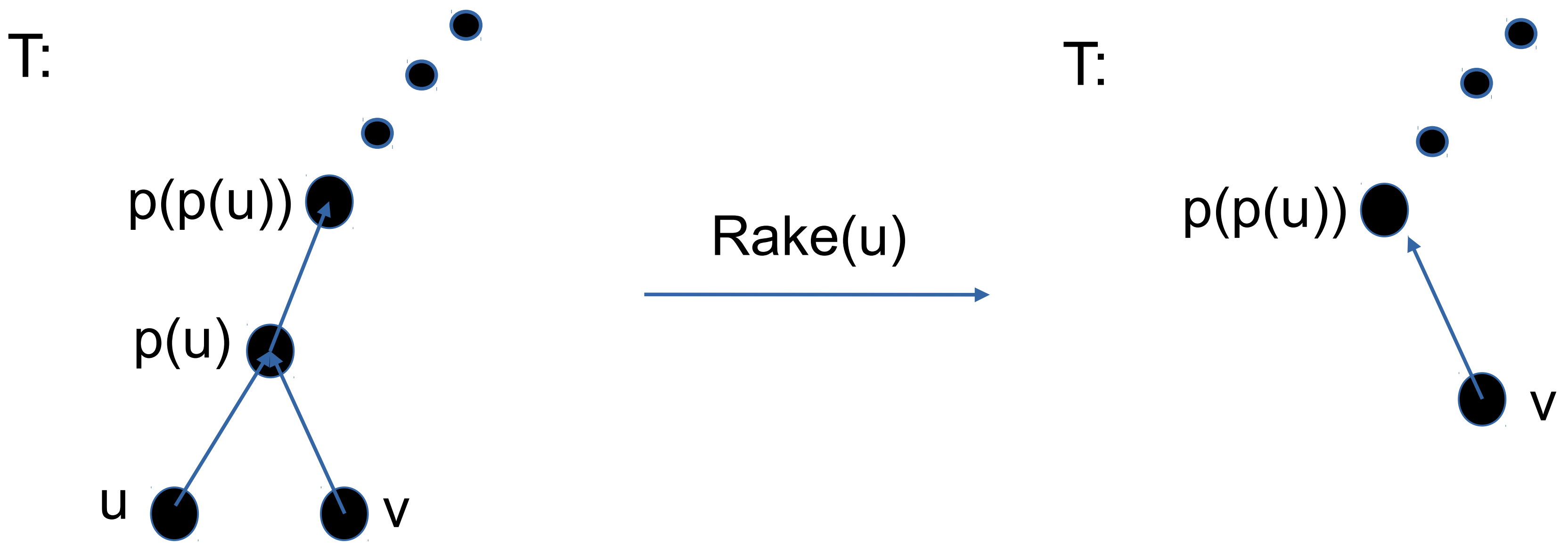
- A RAKE technique that contracts a tree

  rake: 1.an agricultural implement with teeth or tines for gathering cut grass, hay, or the like or for smoothing the surface of the ground.

  2. any of various implements having a similar form, as a croupier's implement for gathering in money on a gaming table.

- Applying the rake technique to evaluate subexpressions.

The Rake Technique



- T = (V, E) be a rooted tree with r as the root and p() be the parent function
- One step of the rake operation at a leaf u with p(u) $\neq$ r involves:
  - Remove nodes u and p(u) from the tree.
  - Make the sibling of u as the child of p(p(u)).

The Rake Technique

- Why is this good?
- Can be applied simultaneously at several leaf nodes in parallel.
- Which ones?
  - All leaves which do not share the same parent, essentially non-siblings.

The Rake Technique

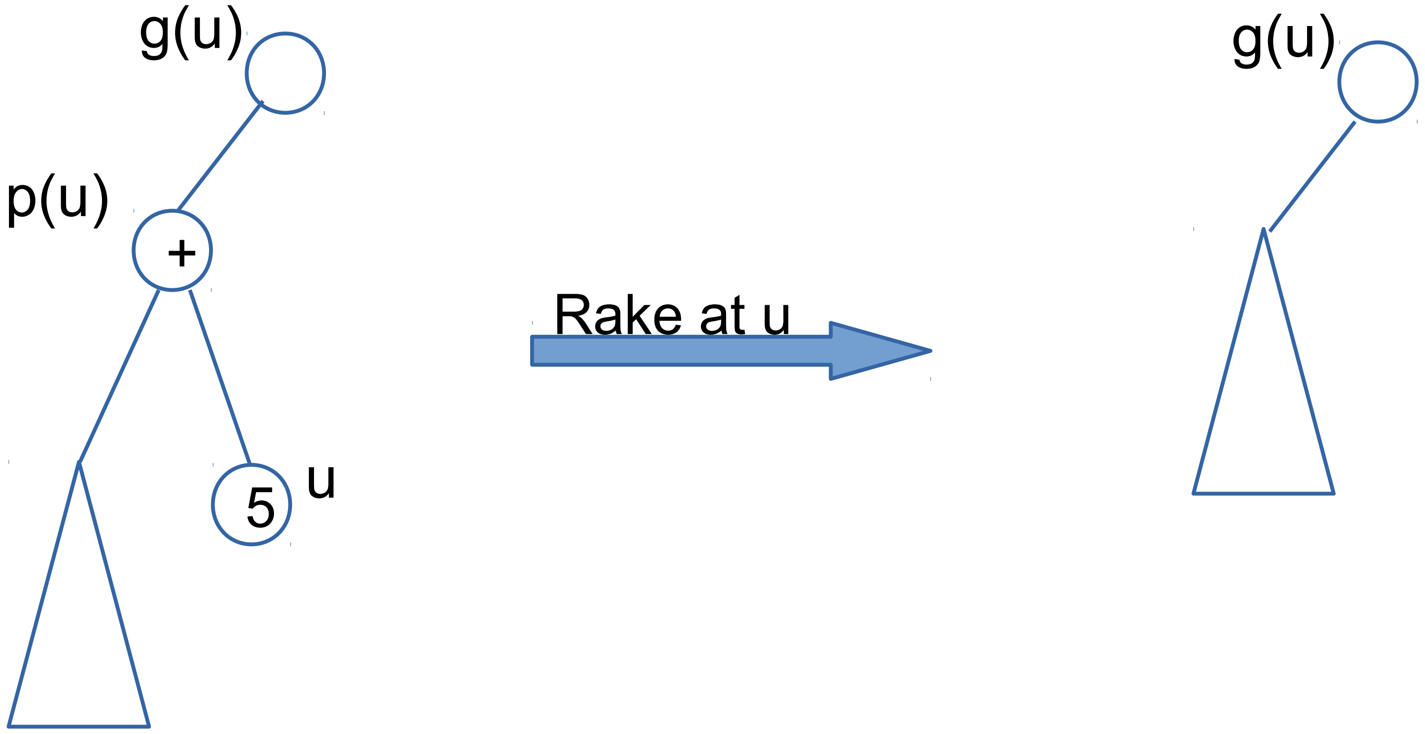
```
Algorithm ShrinkTree(T)
Step 1. Compute labels for the leaf nodes
consecutively, excluding the leftmost and the rightmost
leaf nodes into an array A.
Step 2. for k iterations do
     2.1 Apply the rake operation to all the odd
         numbered leaves that are left children
     2.2 Apply the rake operation to all the odd
            numbered leaves that are right children
     2.3 Update A to be the remaining (even) leaf
nodes.
end-for
End Algorithm.
```

- So, we apply the Rake technique on all non-adjacent sibling nodes.
- The algorithm is as shown.

Time Analysis

- Observation: Each application of the rake at all the leaves as given in the algorithm reduces the number of leaves by a half.
- Each application of Rake at a leaf node is an O(1) operation.
- So the total time is O(log n).
- The number of operations is O(n).
    - Similar observations hold.

Applying Rake to Expression Evaluation

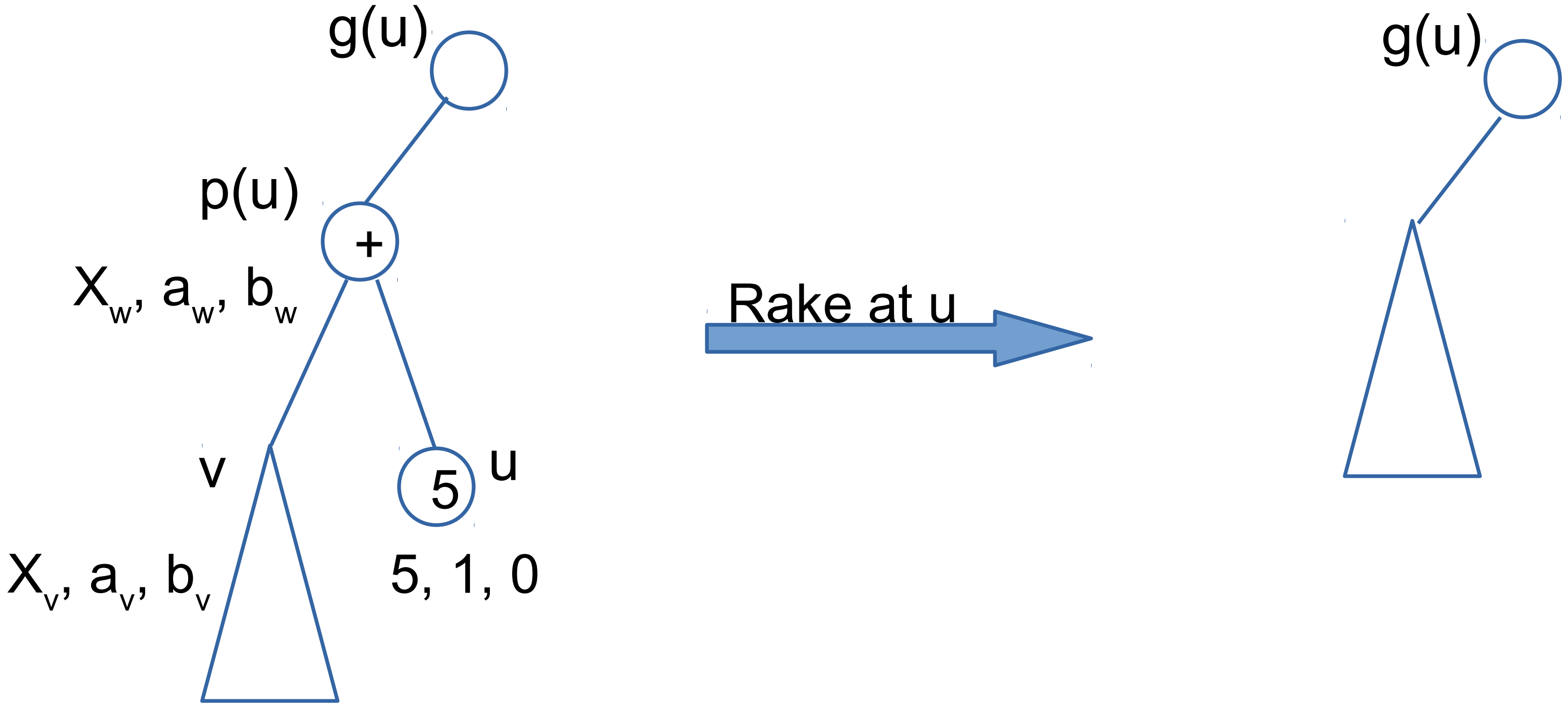


- Applying Rake means that we can process more than one leaf node at the same time.
- For expression evaluation, this may mean that an internal node with only one operand evaluated, also needs to be deleted.
    - Need to partially evaluate internal nodes.

Partial Evaluation

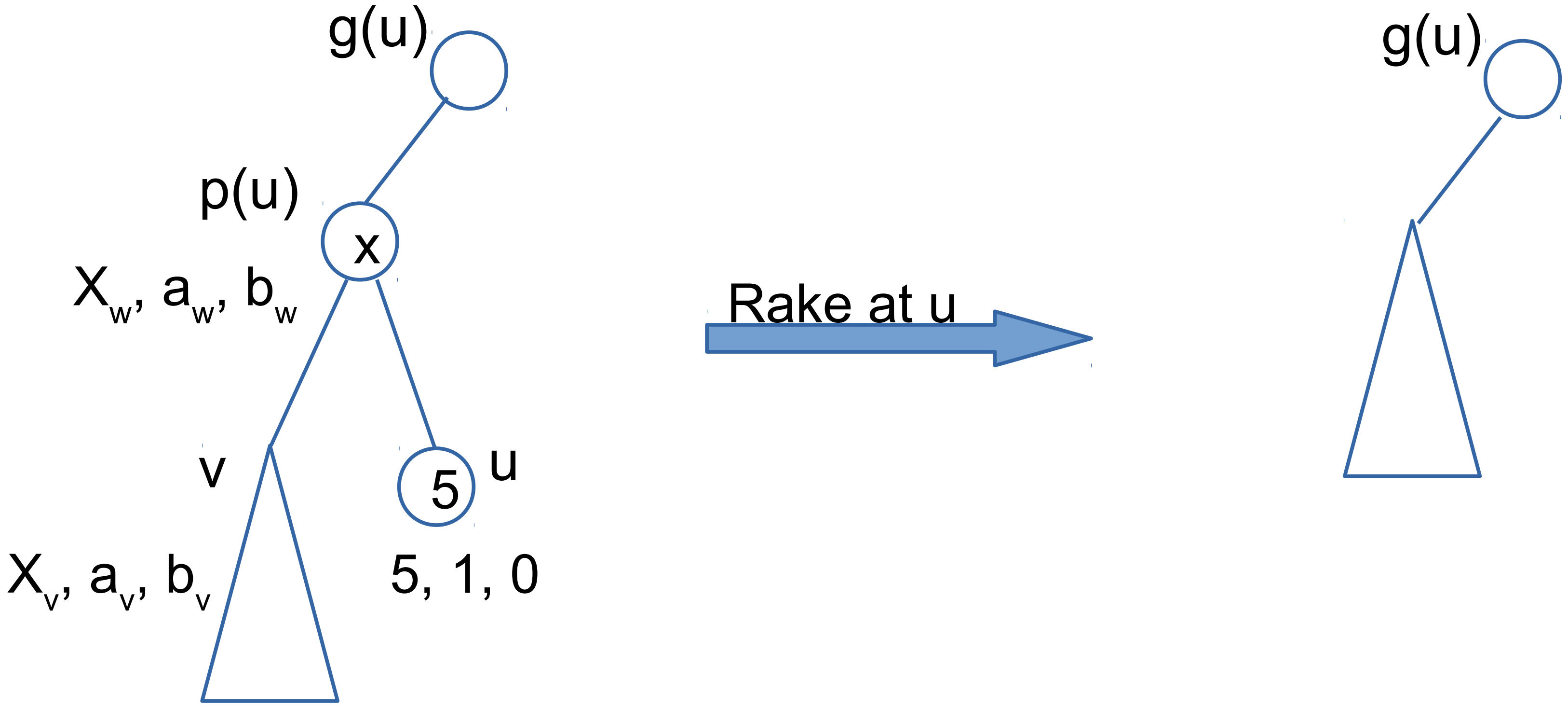
- Transfer the impact of applying the operator at p(u) to the sibling of u.
- Associate with each node u labels a_u and b_u so that $R_u = a_u X_u + b_u$.
- X_u is the result of the subexpression, possibly unknown, at node u.
- Adjust the labels a_u and b_u during any rake operation appropriately.
- Initially, at each leaf node u, X_u equals the operand, $a_u = 1$, and $b_u = 0$.

Adjusting Labels



- Prior to rake at u, contribution of p(u) to g(u) is $a_w X_w + b_w$.
- $X_w = (a_u X_u + b_u) + (a_v X_v + b_v) = a_v X_v + (a_u X_u + b_u + b_v)$
- Therefore, adjust a_v and b_v as $a_w a_v$ and $a_w(a_u X_u + b_u + b_v)$.

Adjusting Labels



- Prior to rake at u, contribution of p(u) to g(u) is $a_w X_w + b_w$.
- $X_w = (a_u X_u + b_u) \times (a_v X_v + b_v)$
- Therefore, adjust a_v and b_v as:
- $a_v = a_w a_v (a_u X_u + b_u)$, $b_v = b_w + a_w b_v (a_u X_u + b_u)$.

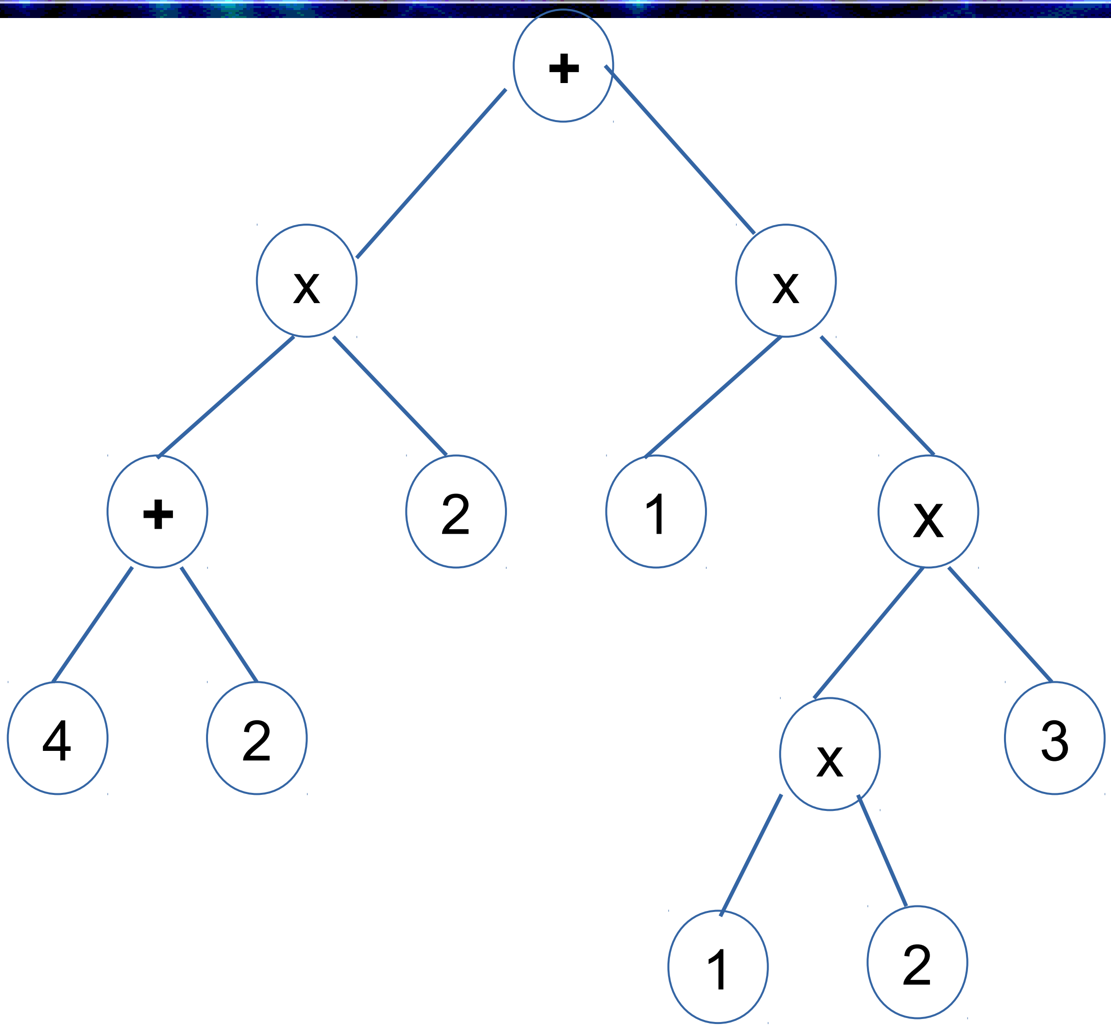
Adjusting Labels

For other operators, proceed in a similar fashion.
HW Problem.

Expression Evaluation

- Parallel algorithm has the following main steps:
  - Rake the expression tree
  - Set up and adjust labels while raking
  - Stop when the tree has only three nodes, one operator and two operands as children.
  - Evaluate this three node tree.
- Theorem: Expression evaluation of an n-node expression tree can be done in parallel on an EREW PRAM using O(log n) time and O(n) operations.

Example


+
x
x
+
2
1
x
4
2
x
3
1
2