CSC 225

Algorithms and Data Structures I

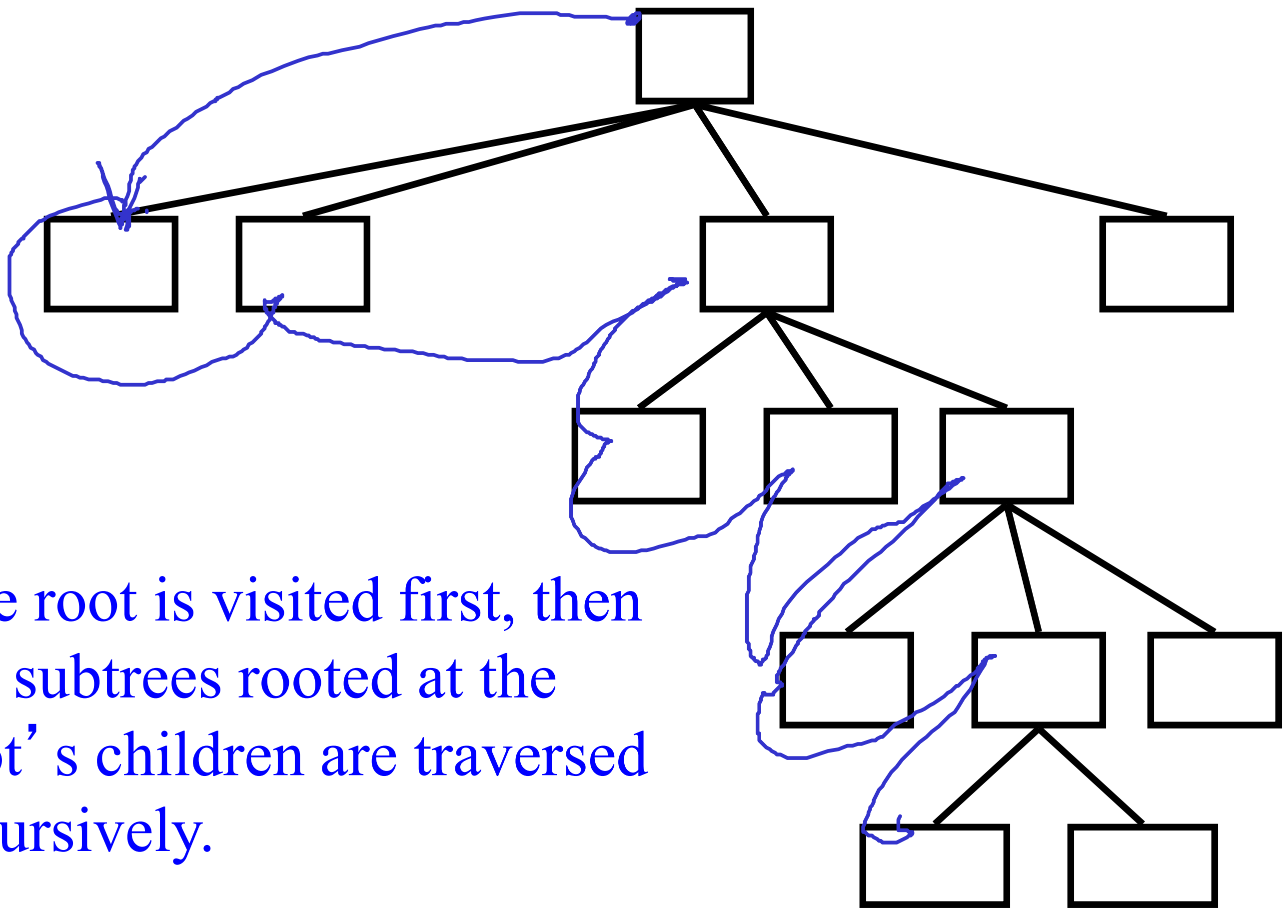
Rich Little
rlittle@uvic.ca
ECS 516

Tree Traversals

- n-ary tree traversals
  - Preorder
  - Postorder
  - Level order

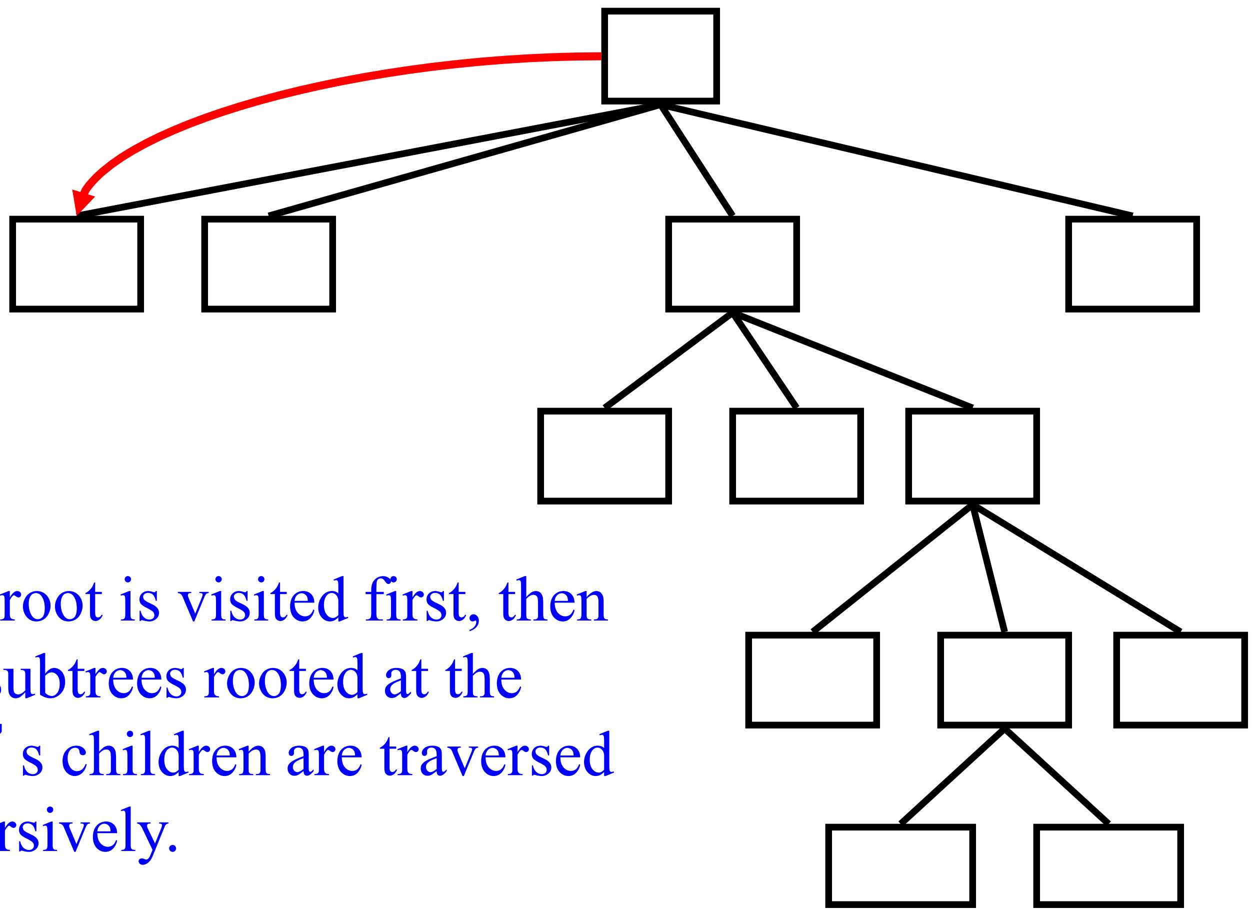
- Binary tree traversals
  - Preorder
  - Postorder
  - Inorder
  - Level order

Preorder Traversal
Depth First Search



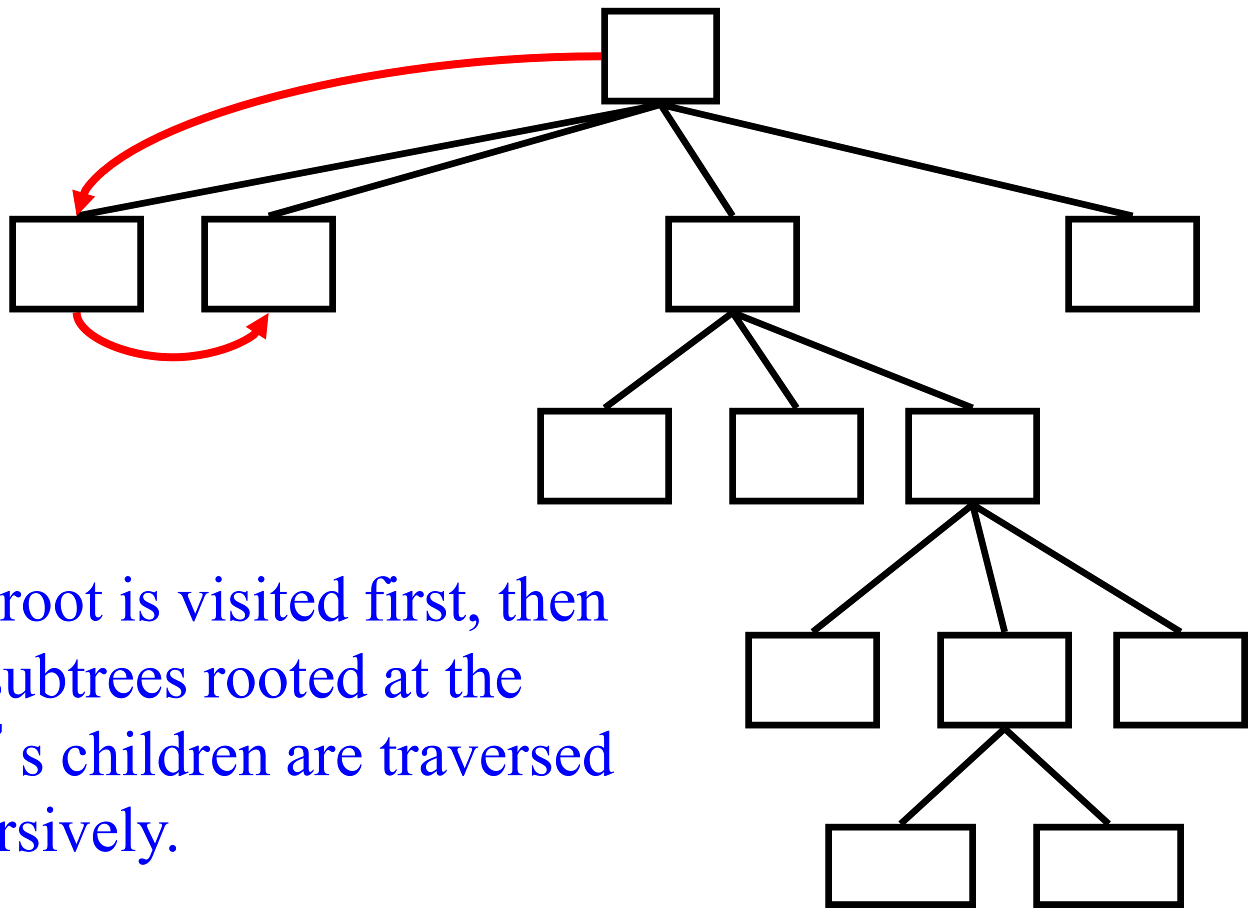
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



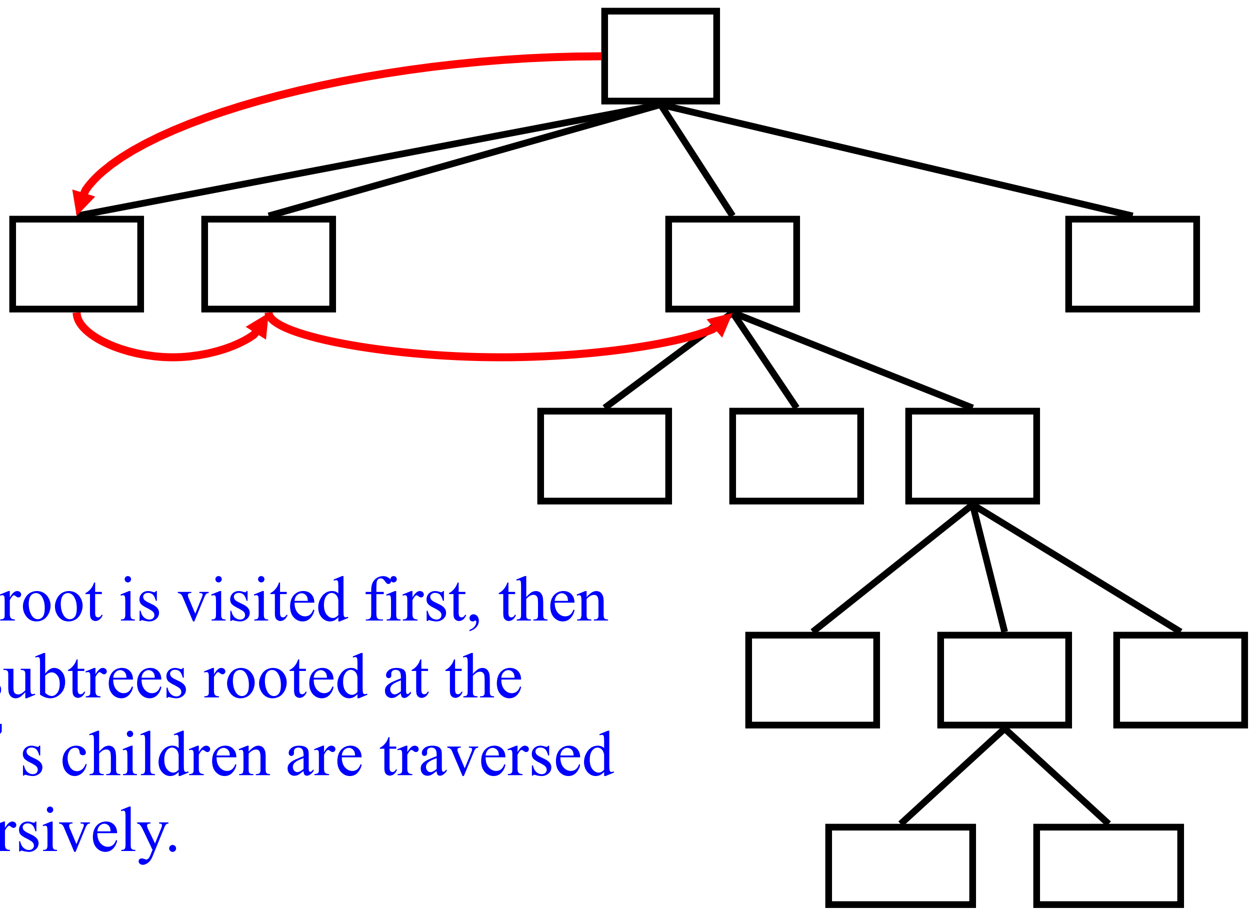
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



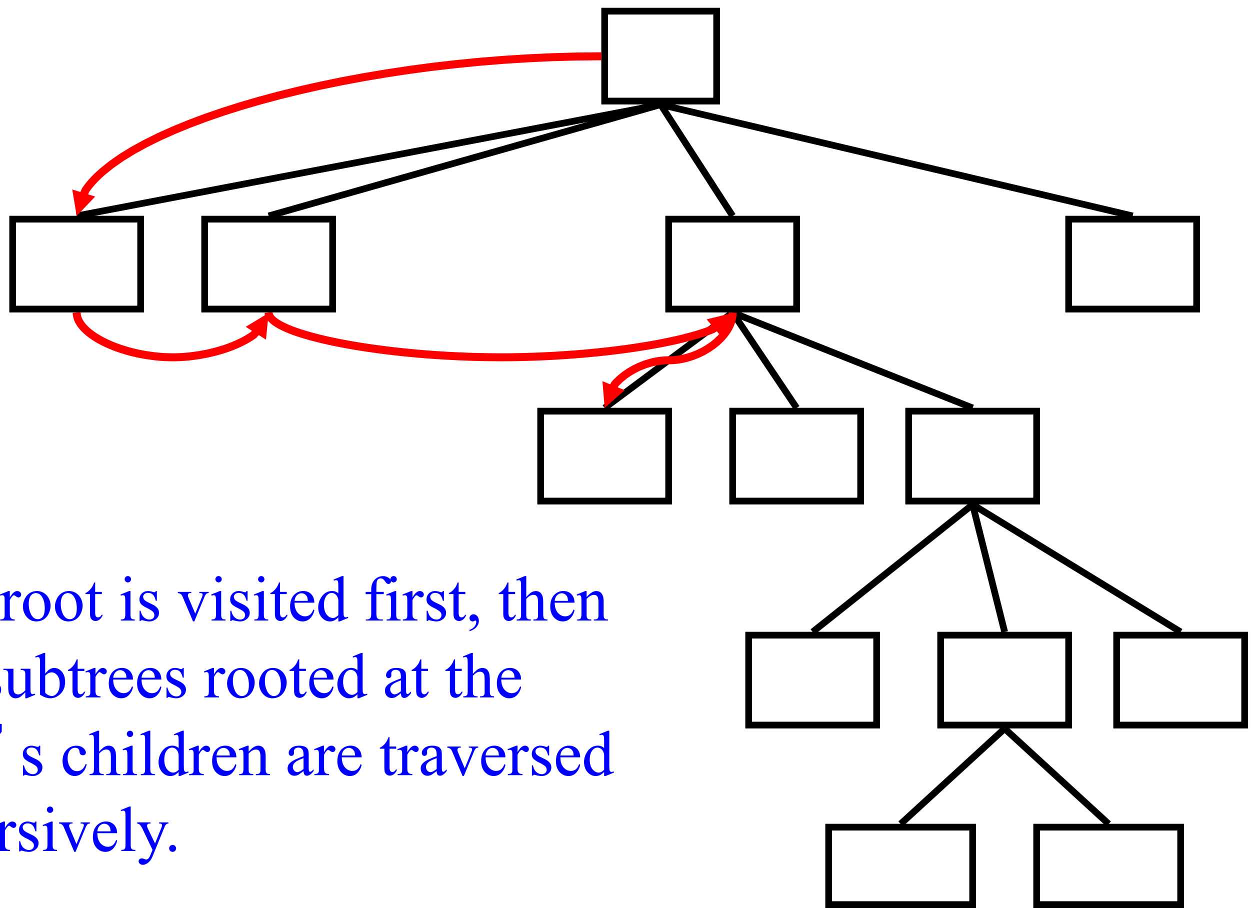
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



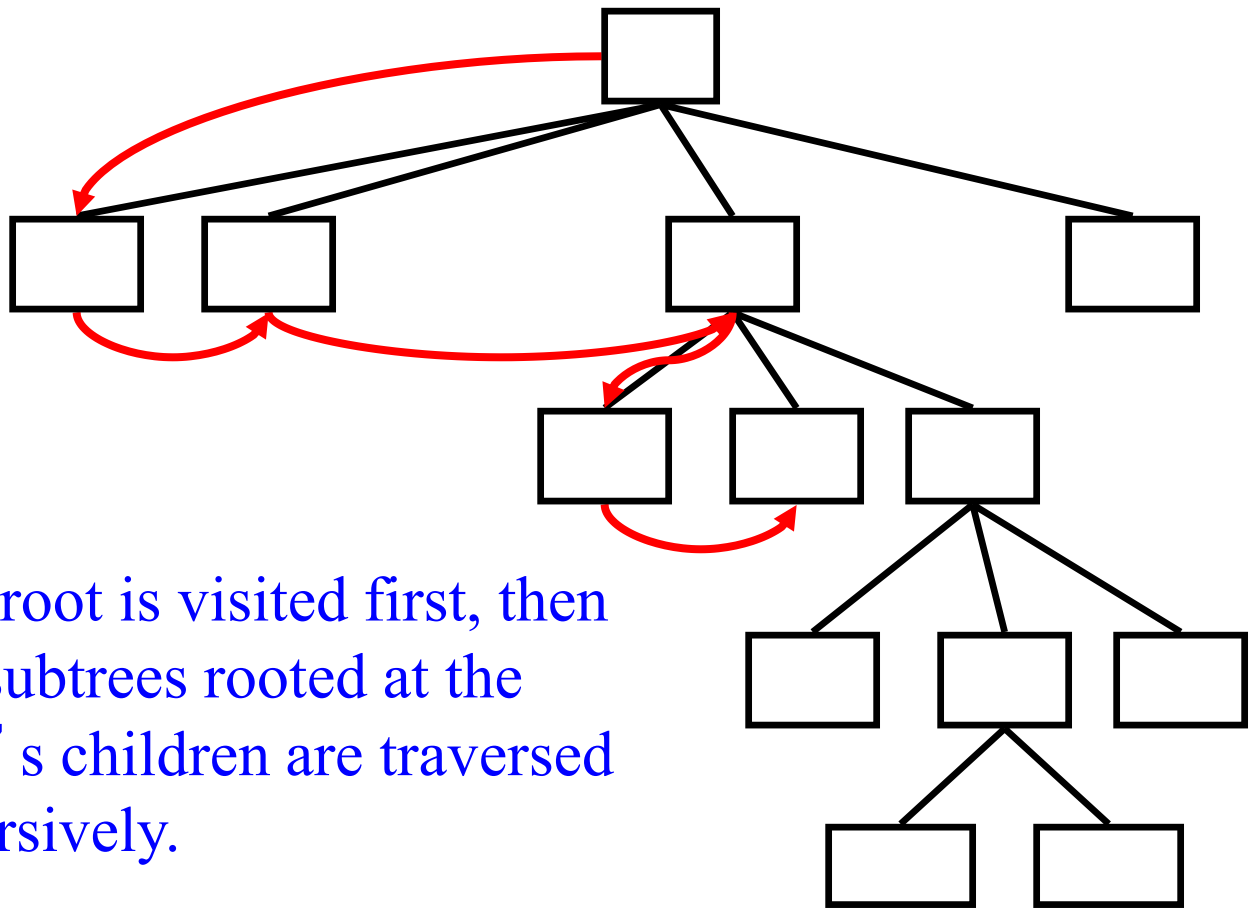
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



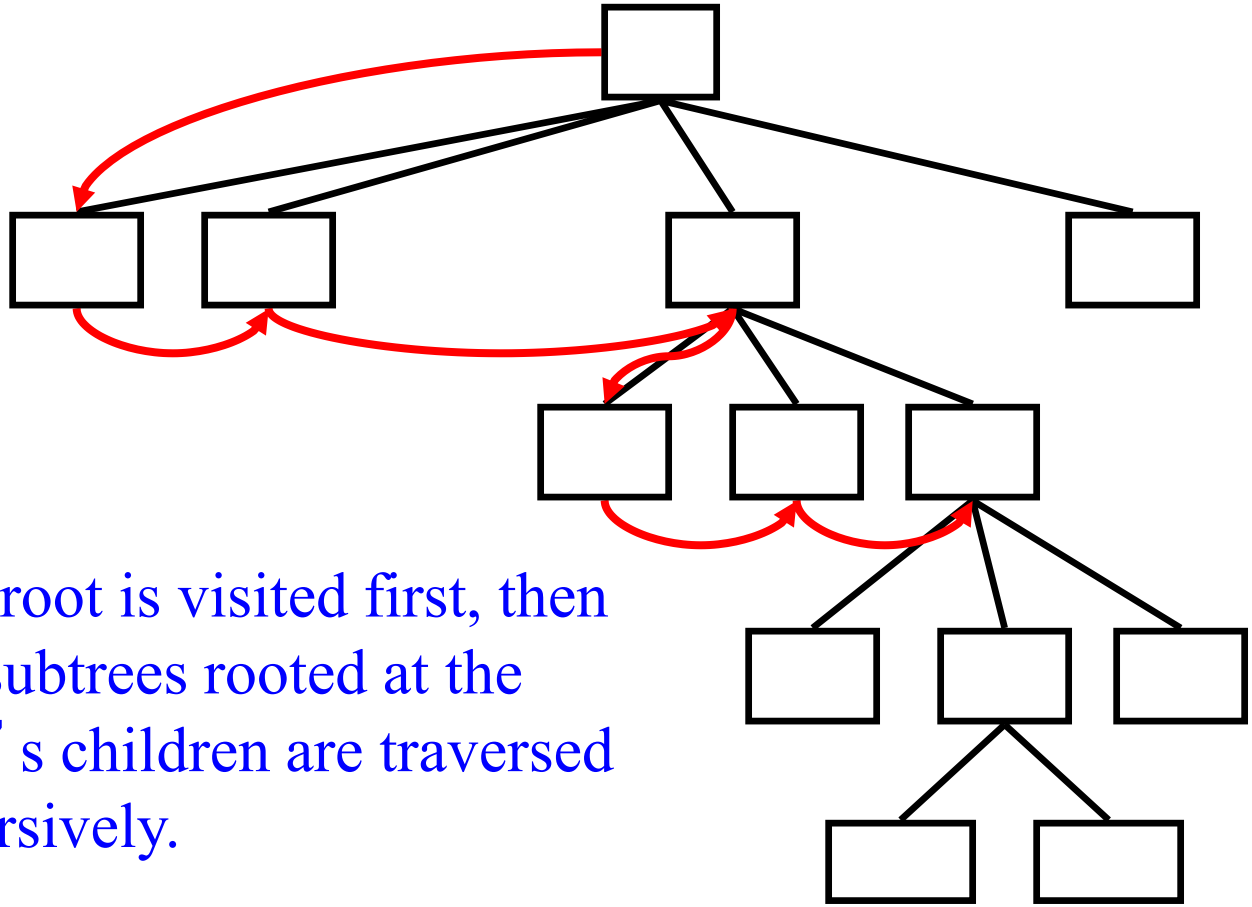
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



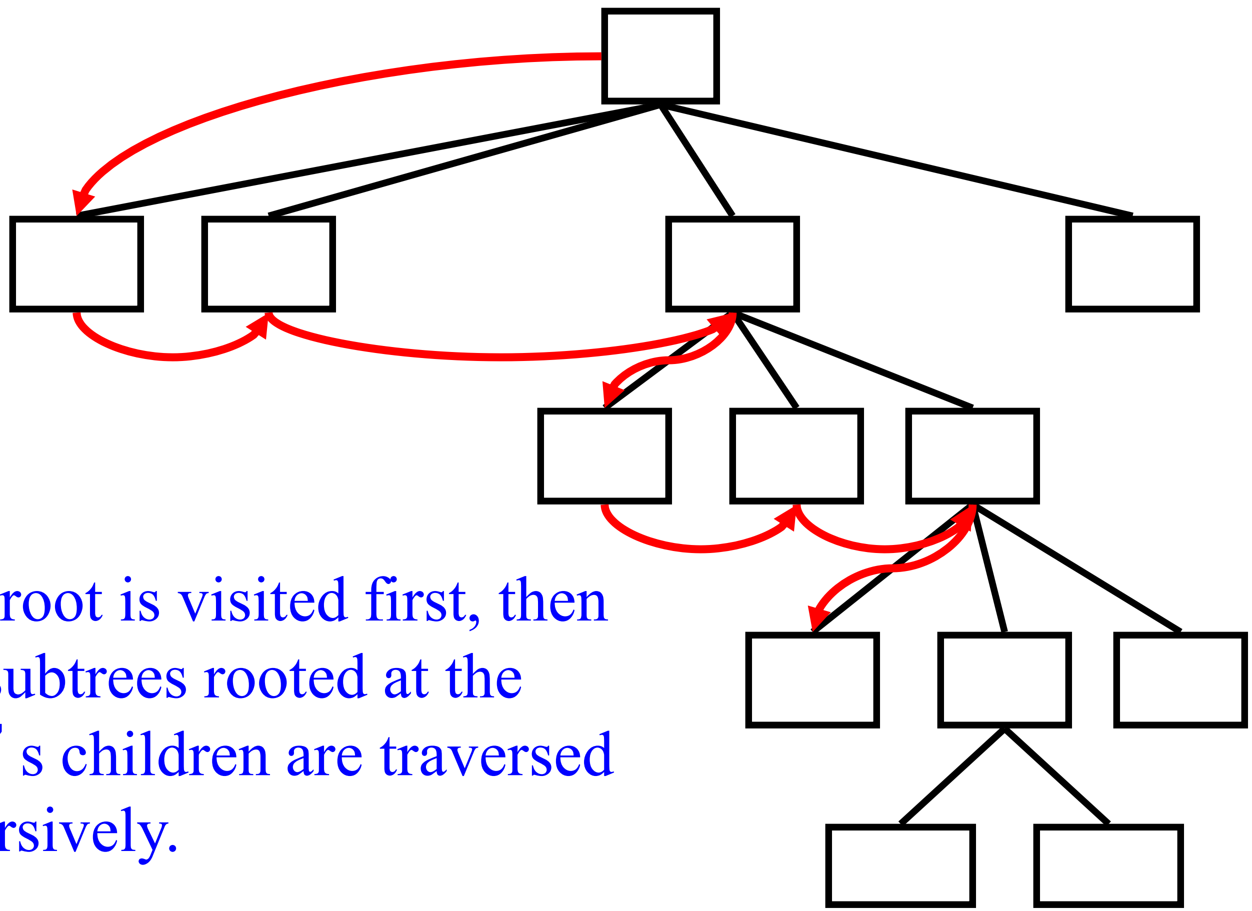
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



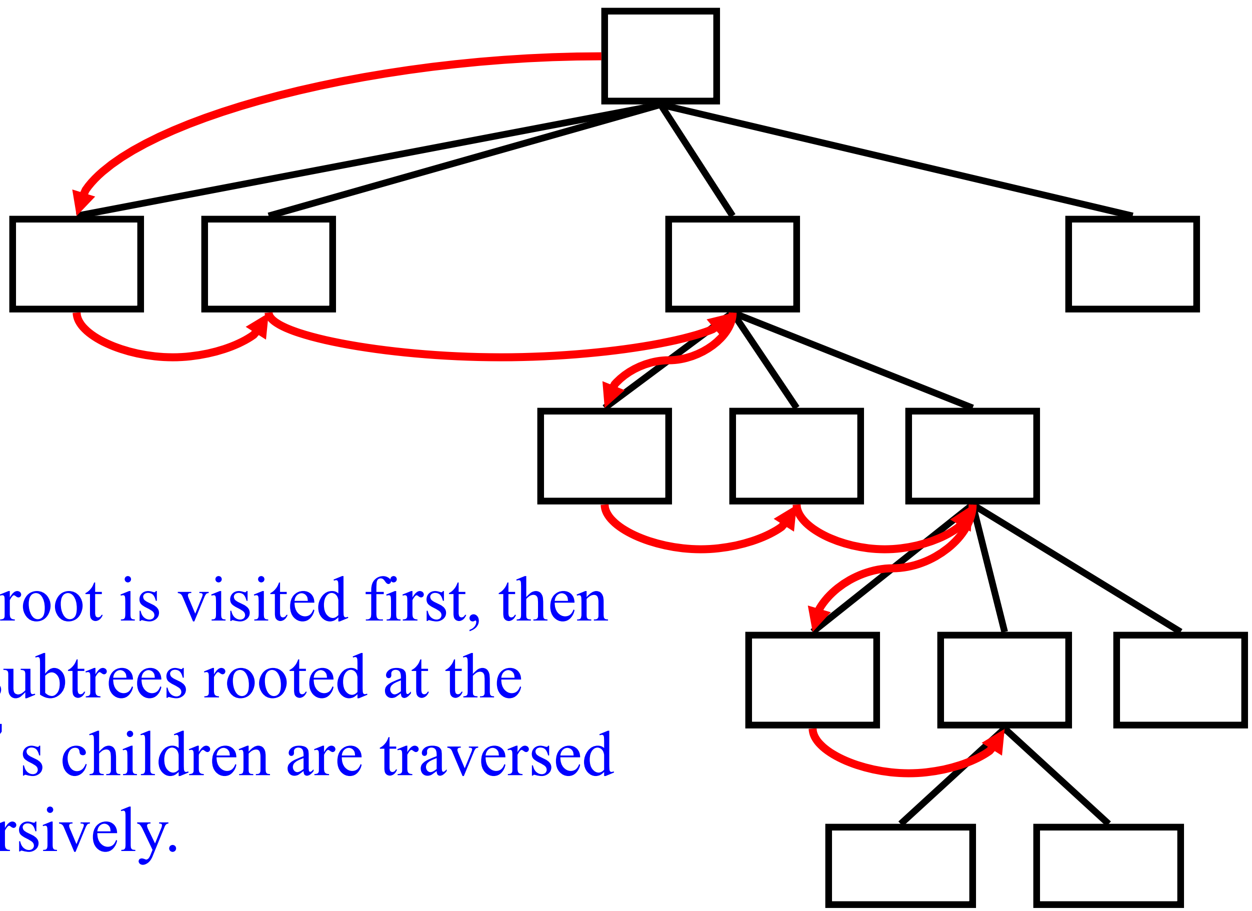
The root is visited first, then the subtrees rooted at the root' s children are traversed recursively.

Preorder Traversal
Depth First Search



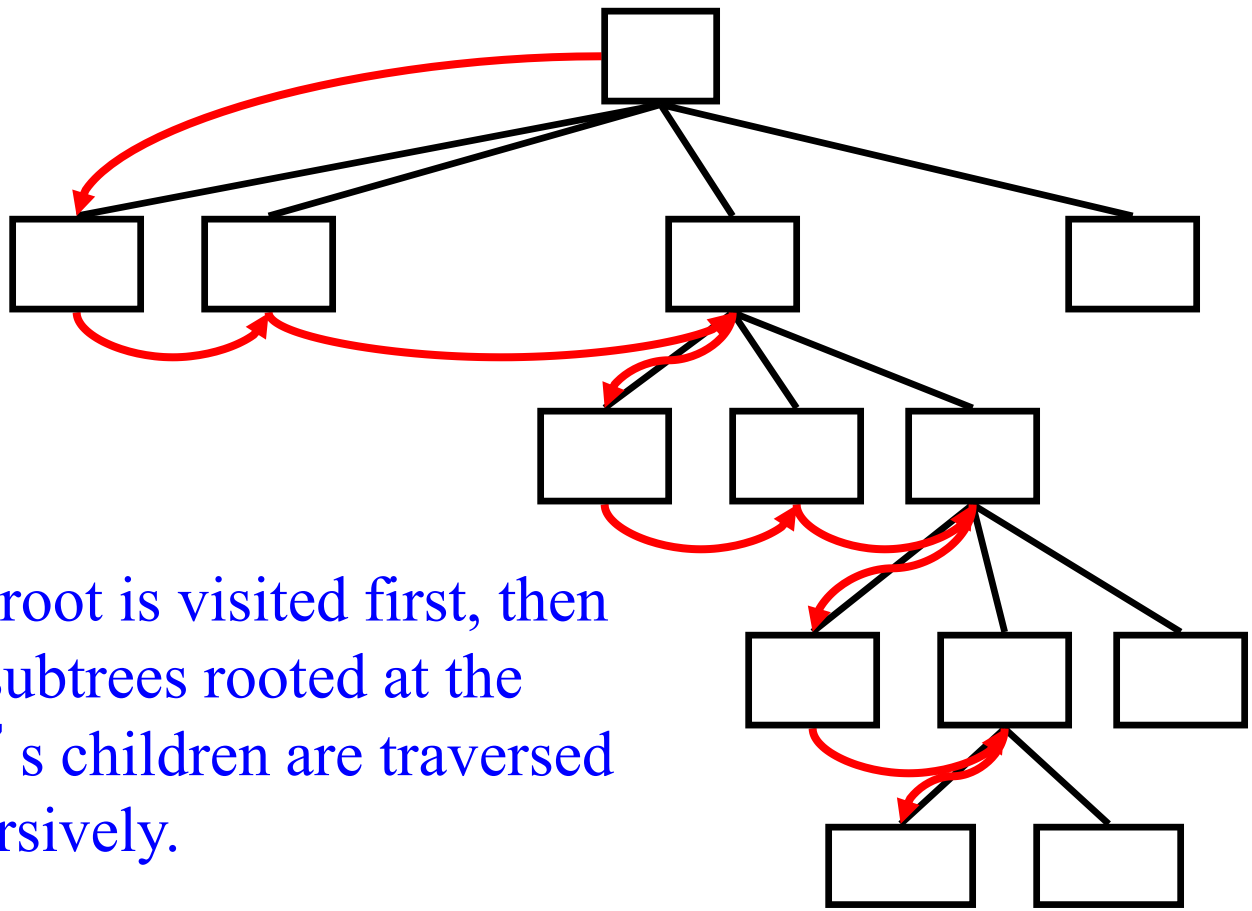
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



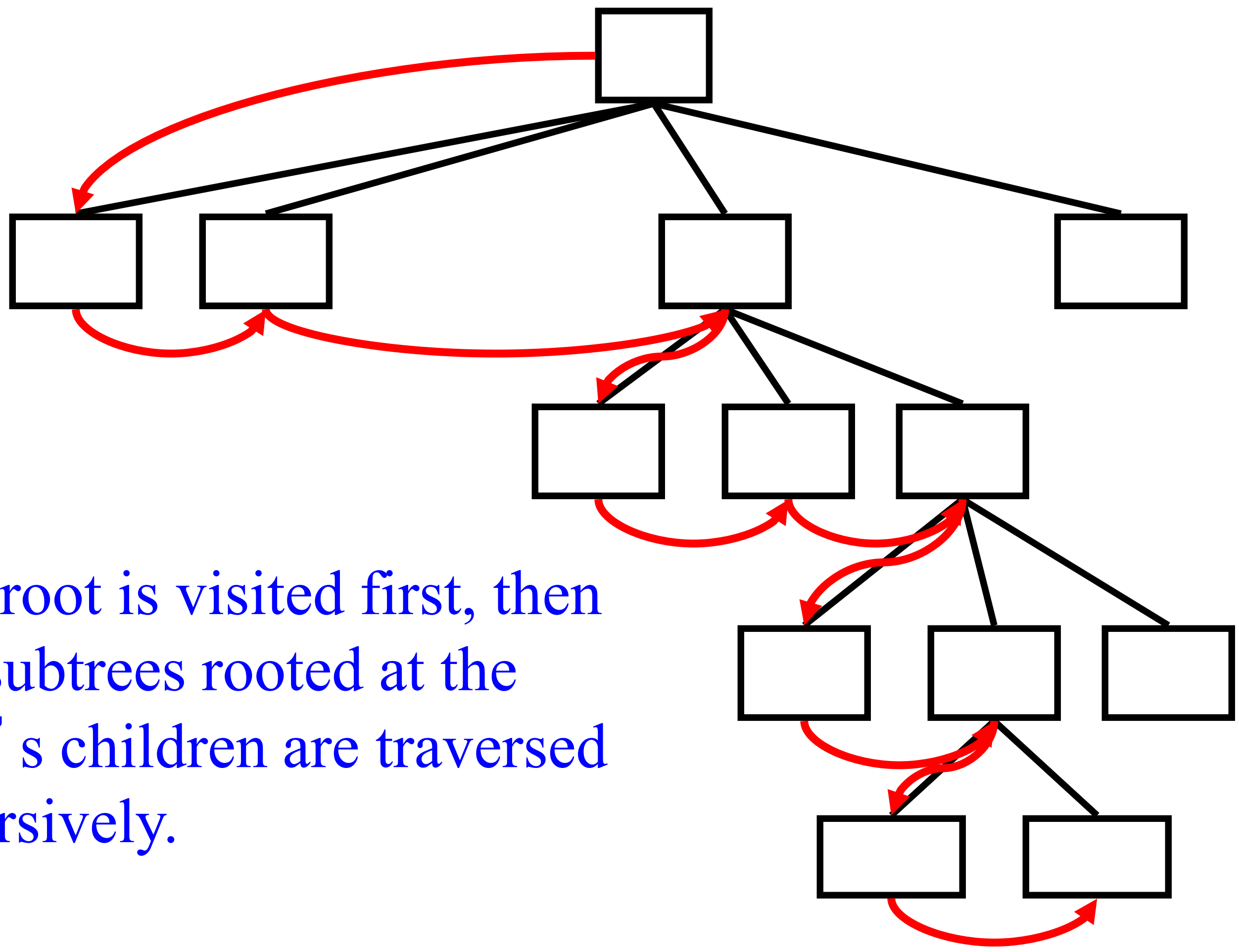
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



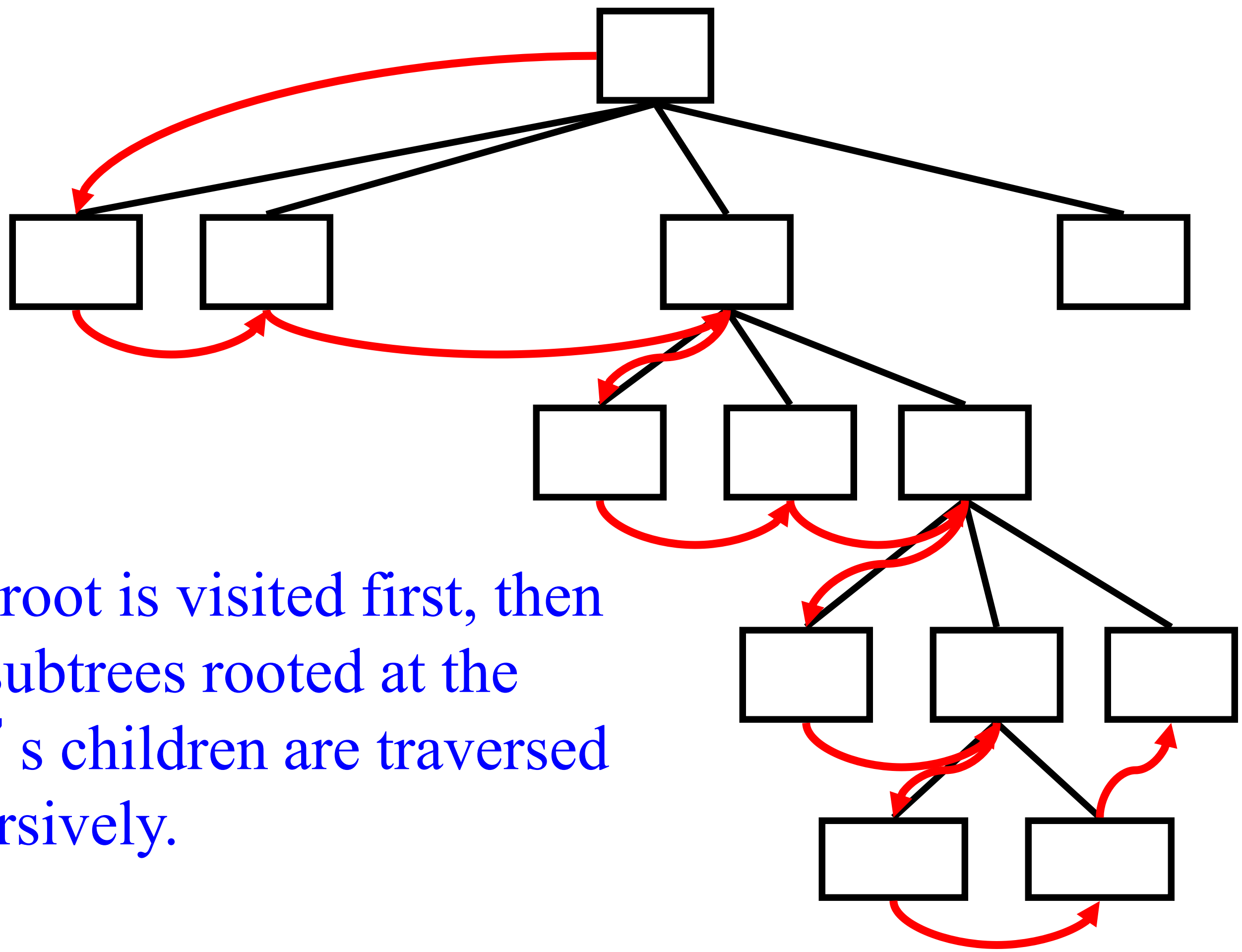
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



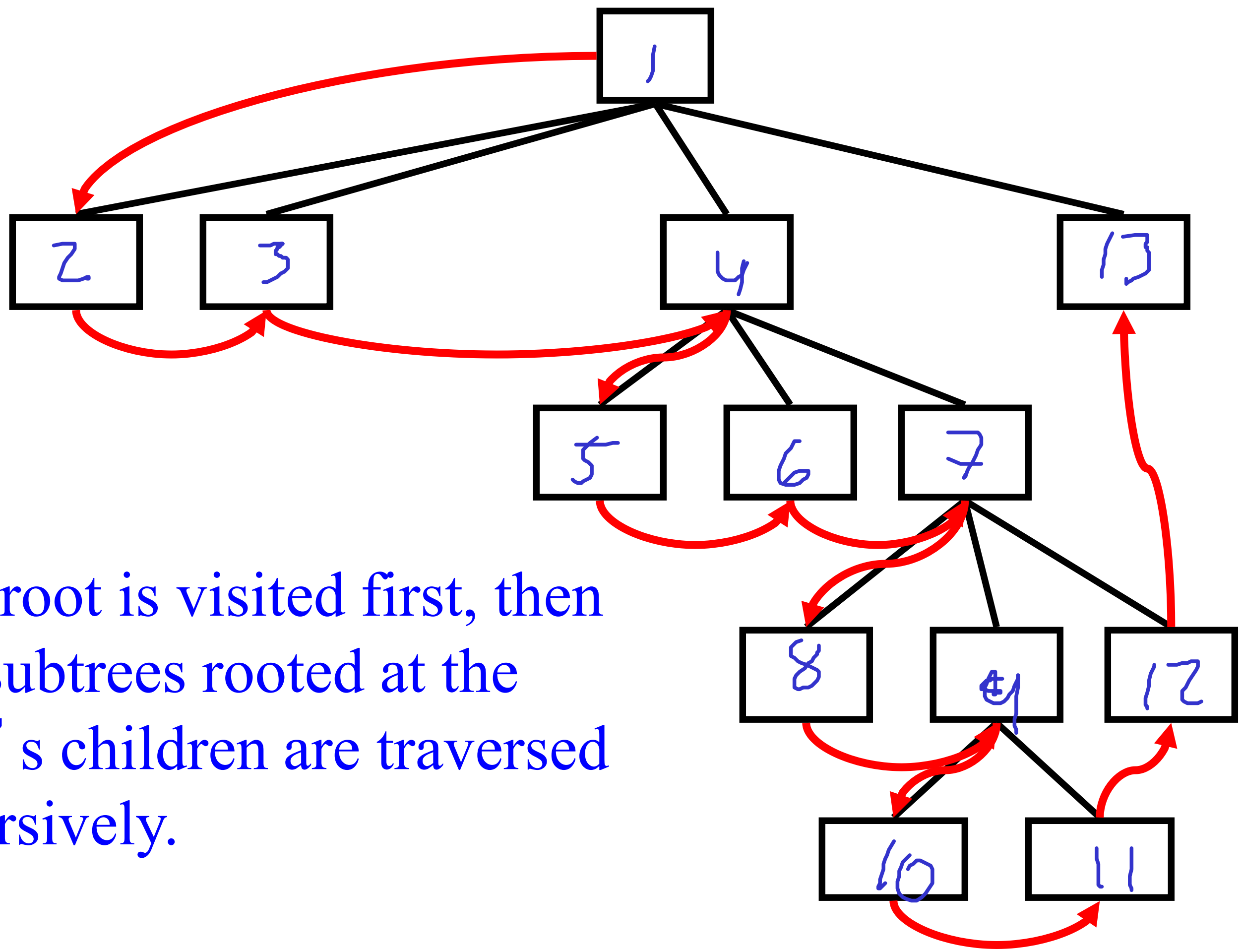
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



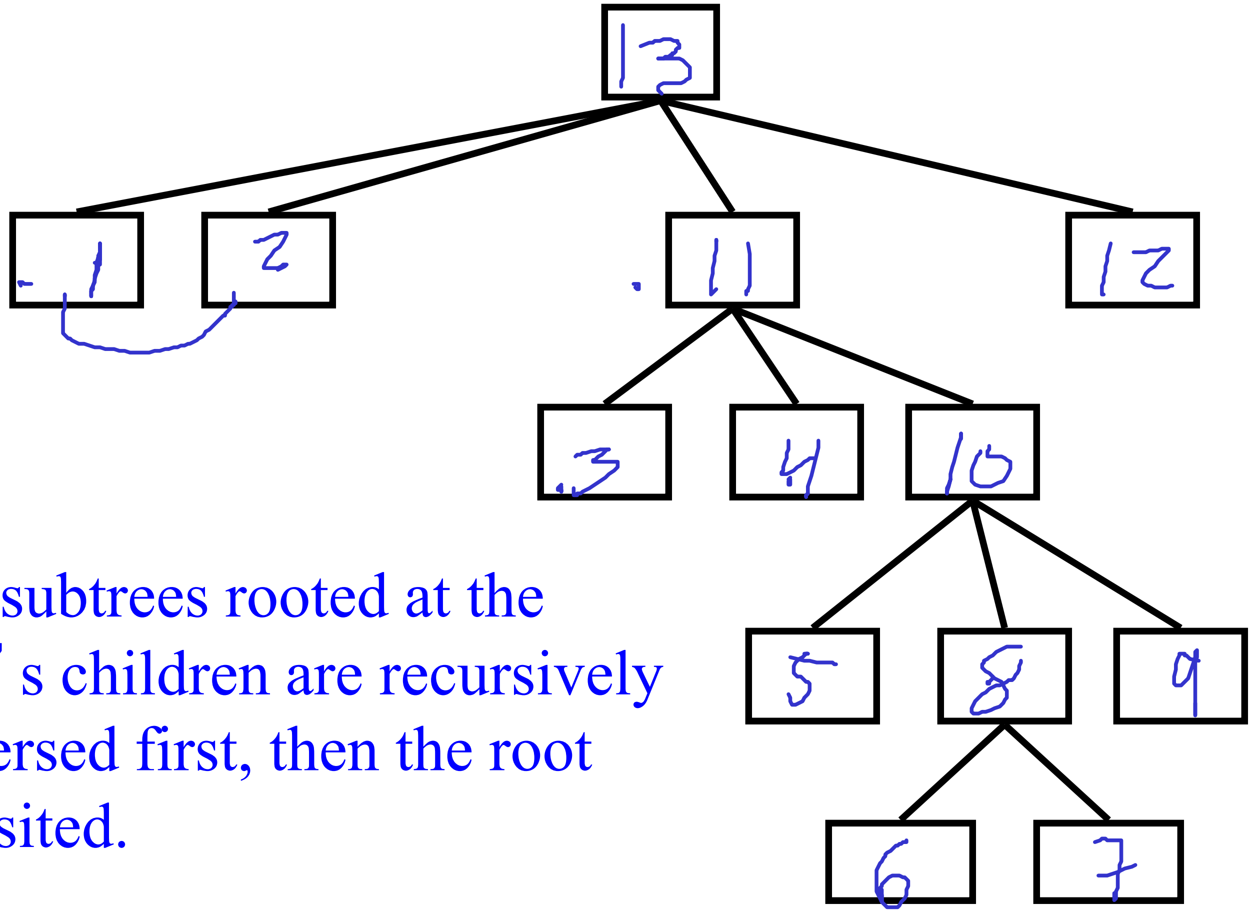
The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

Preorder Traversal
Depth First Search



The root is visited first, then the subtrees rooted at the root's children are traversed recursively.

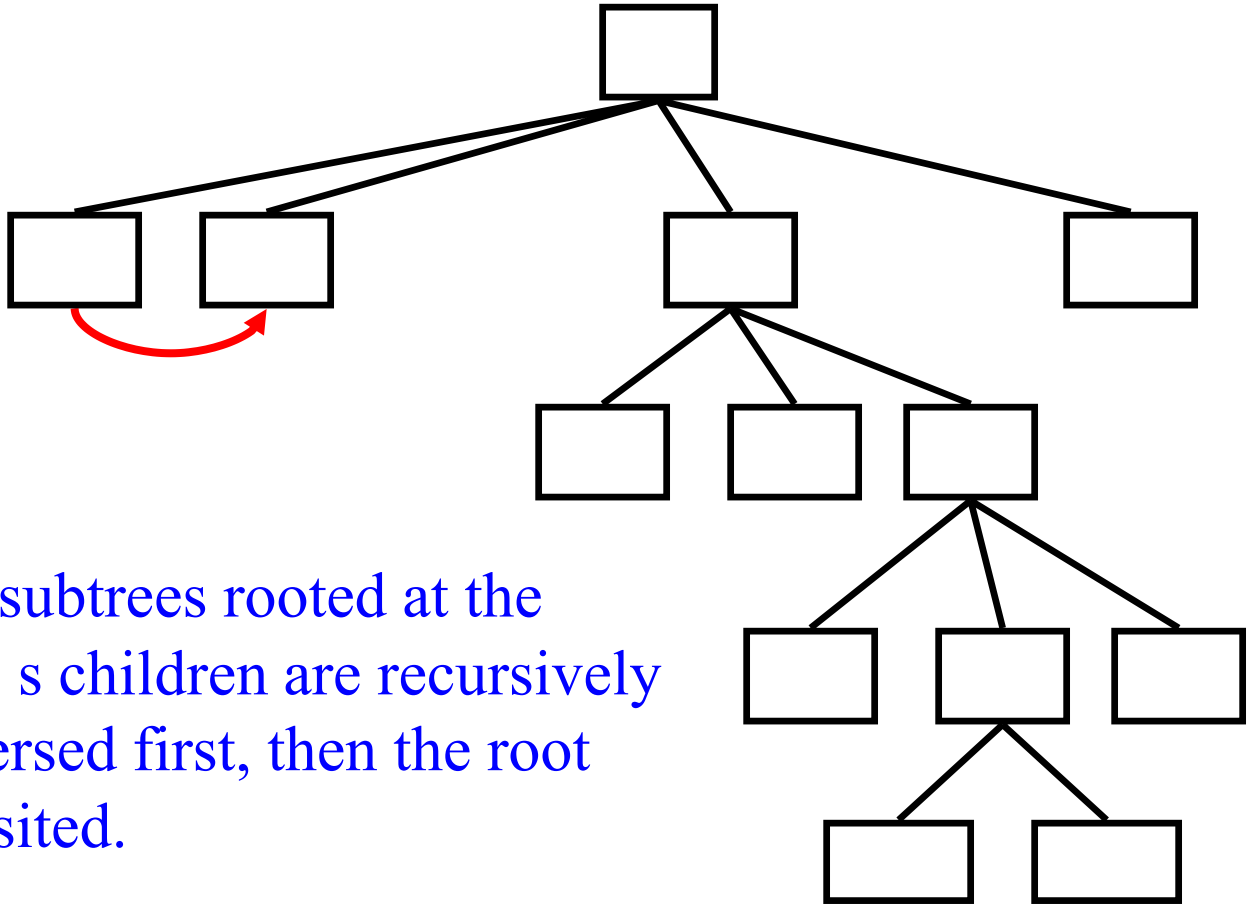
Postorder Traversal



The subtrees rooted at the root’s children are recursively traversed first, then the root is visited.

.

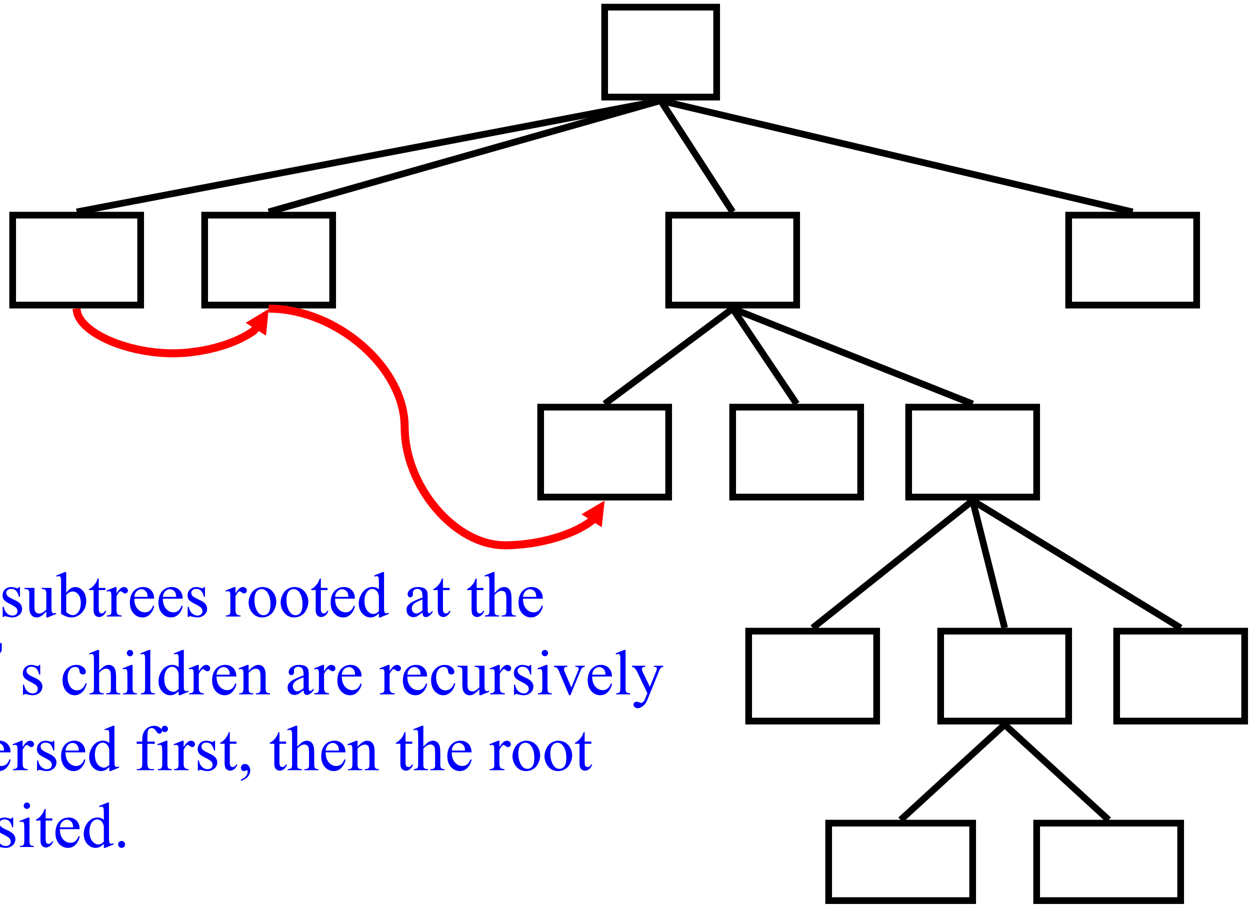
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

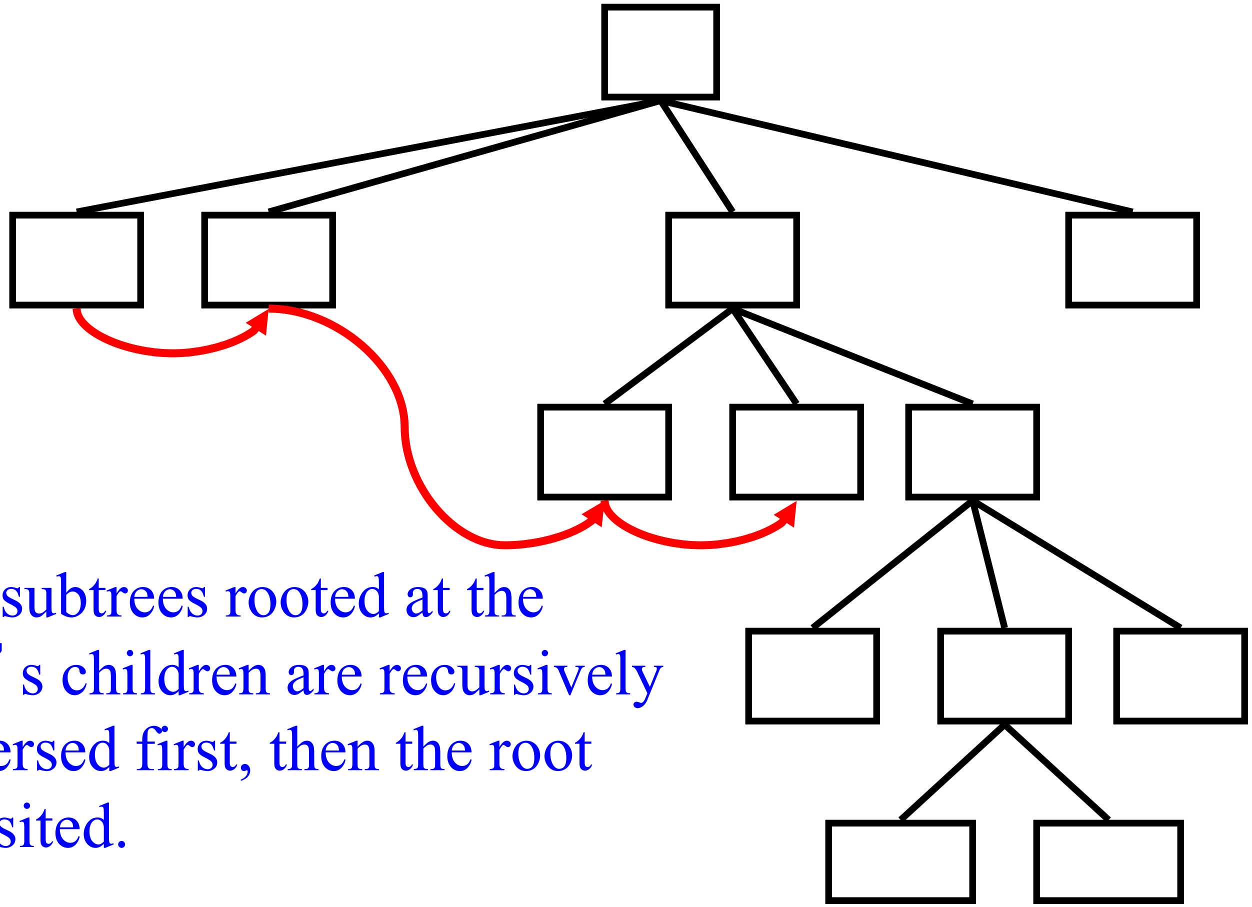
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

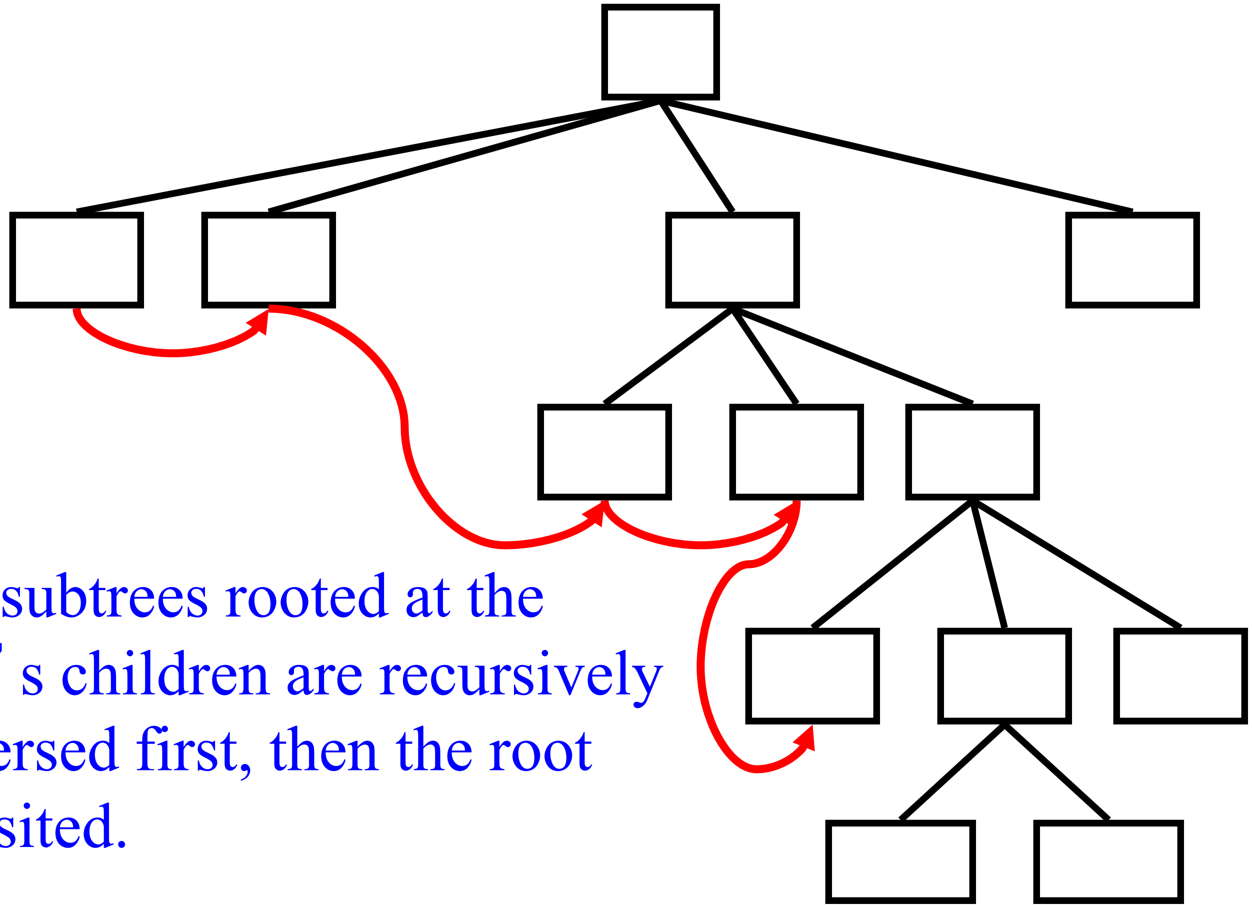
Postorder Traversal



The subtrees rooted at the root’s children are recursively traversed first, then the root is visited.

.

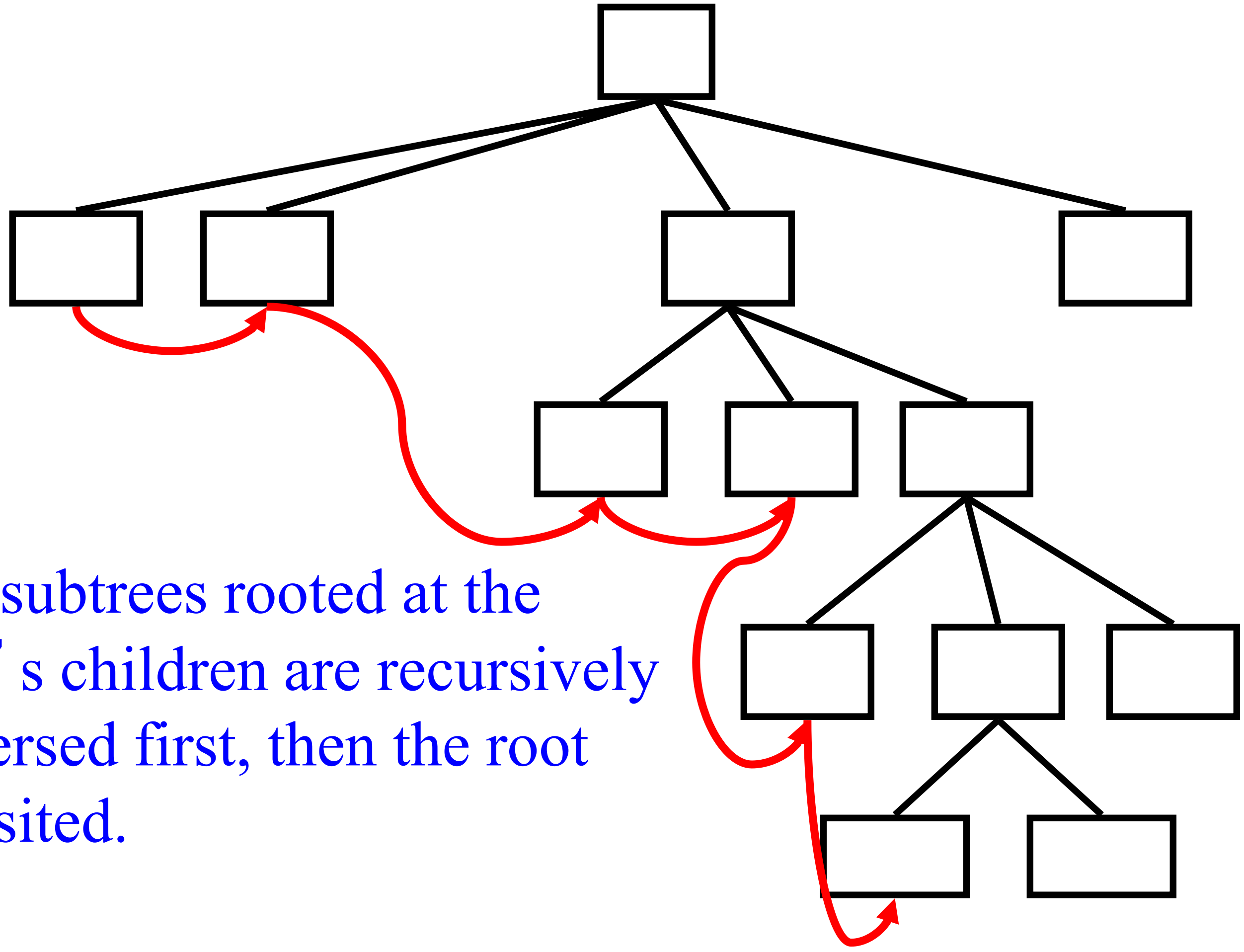
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

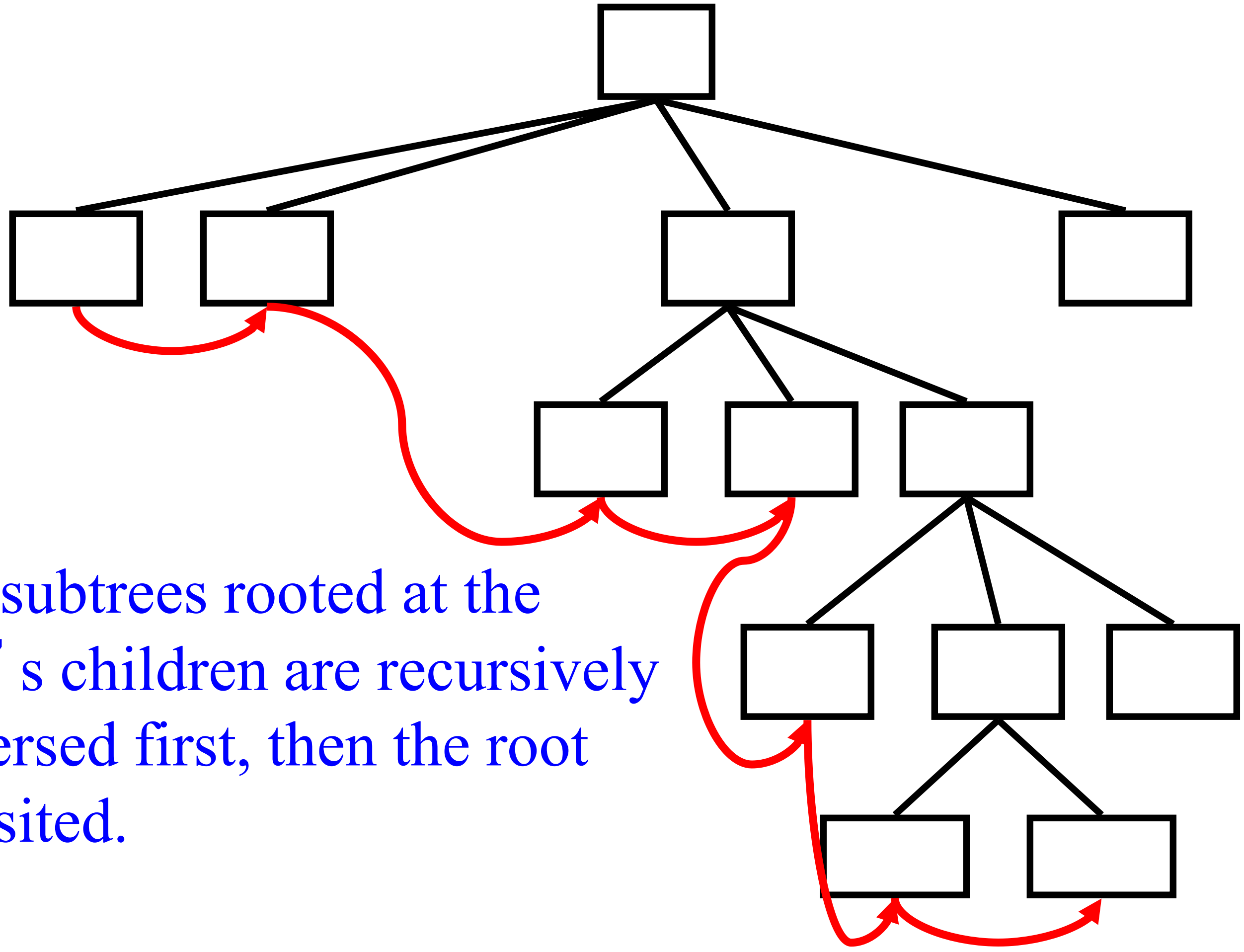
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

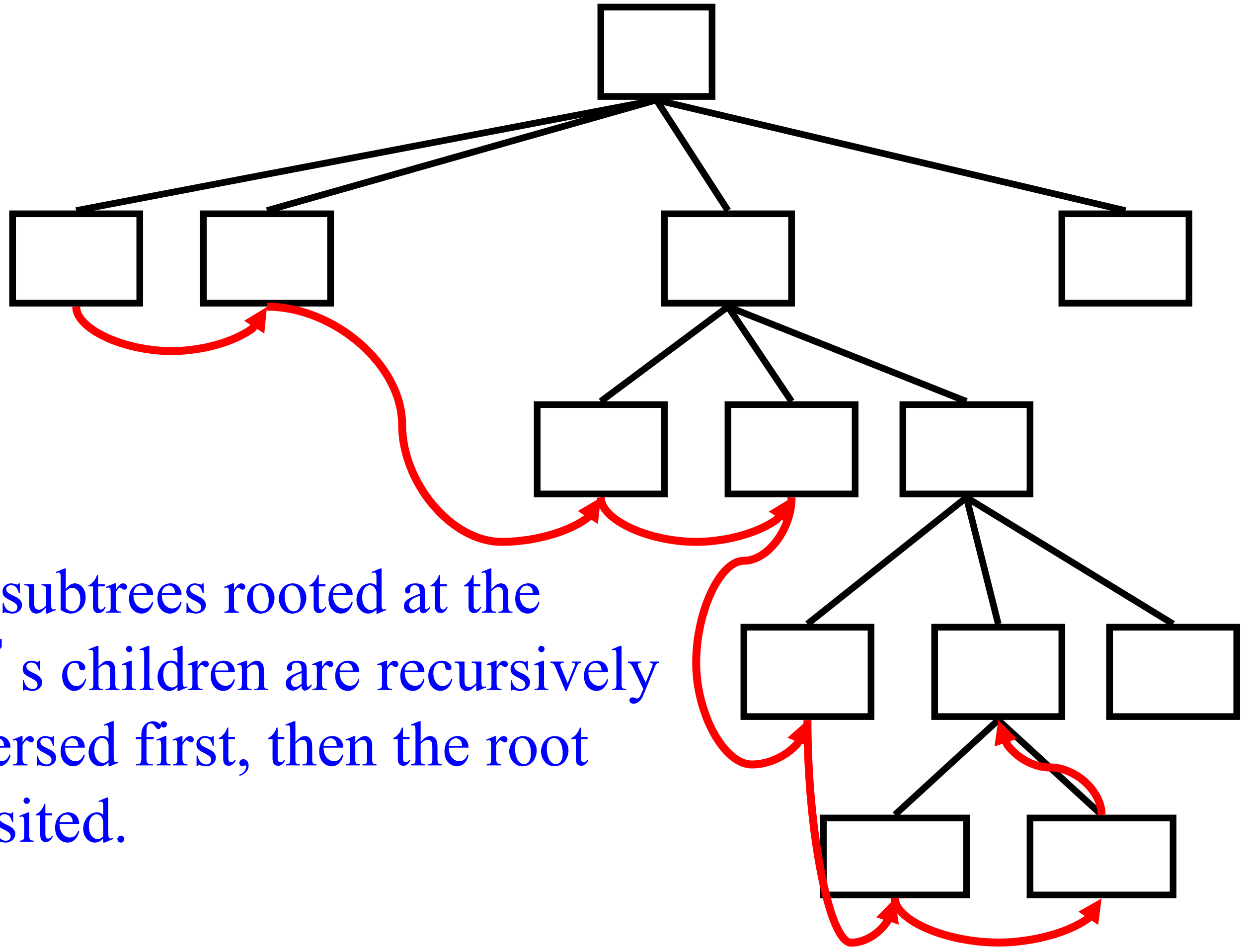
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

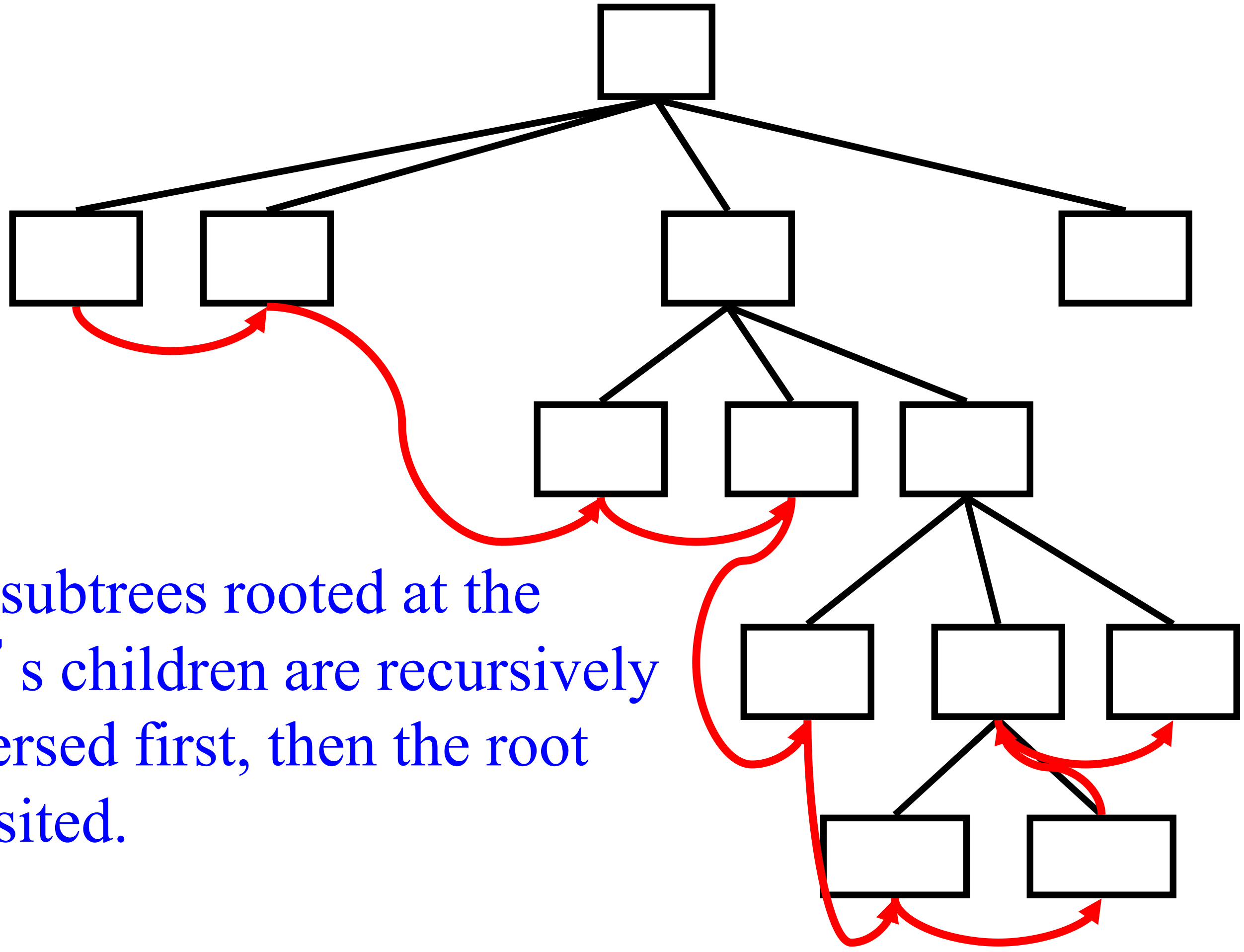
Postorder Traversal



The subtrees rooted at the root’s children are recursively traversed first, then the root is visited.

.

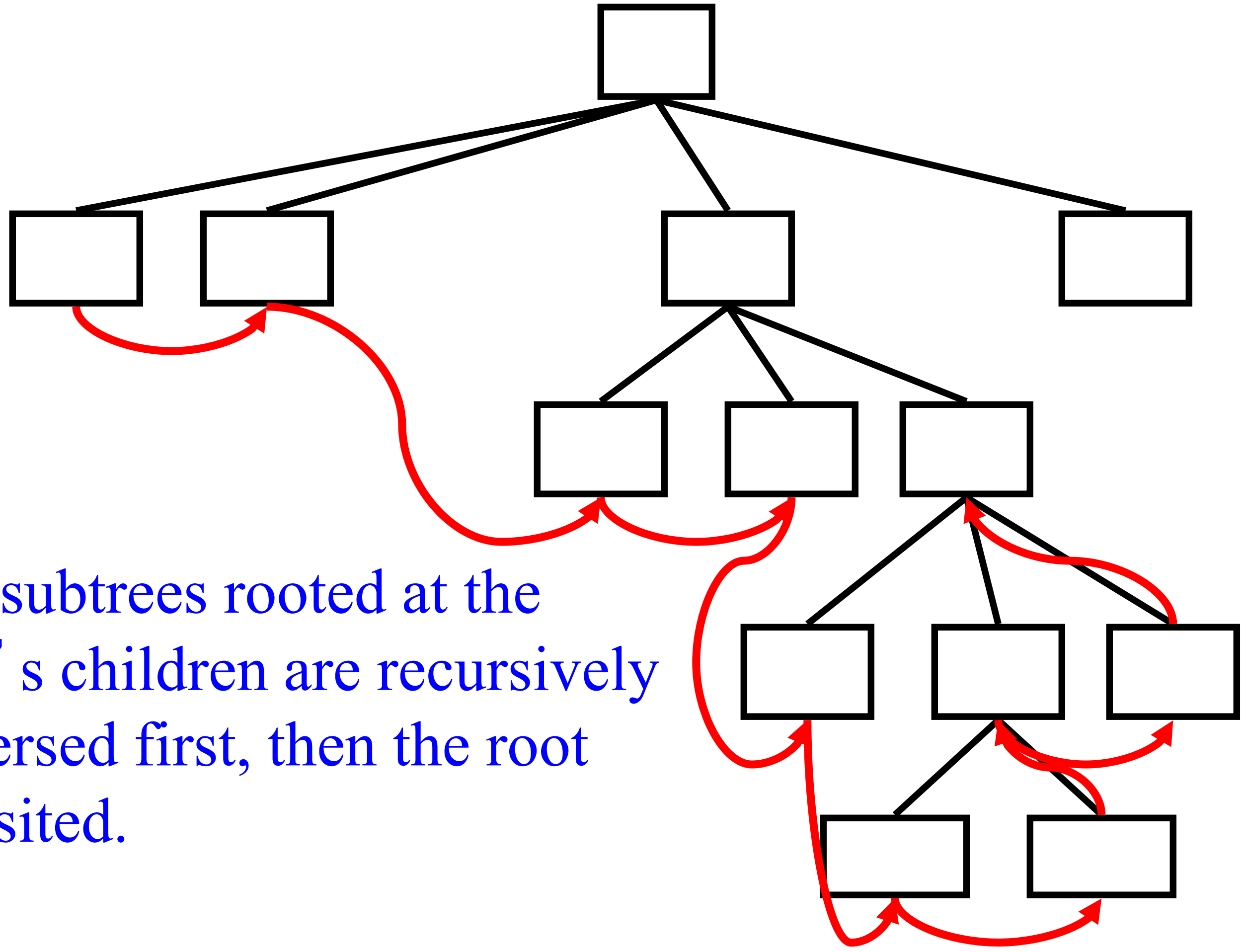
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

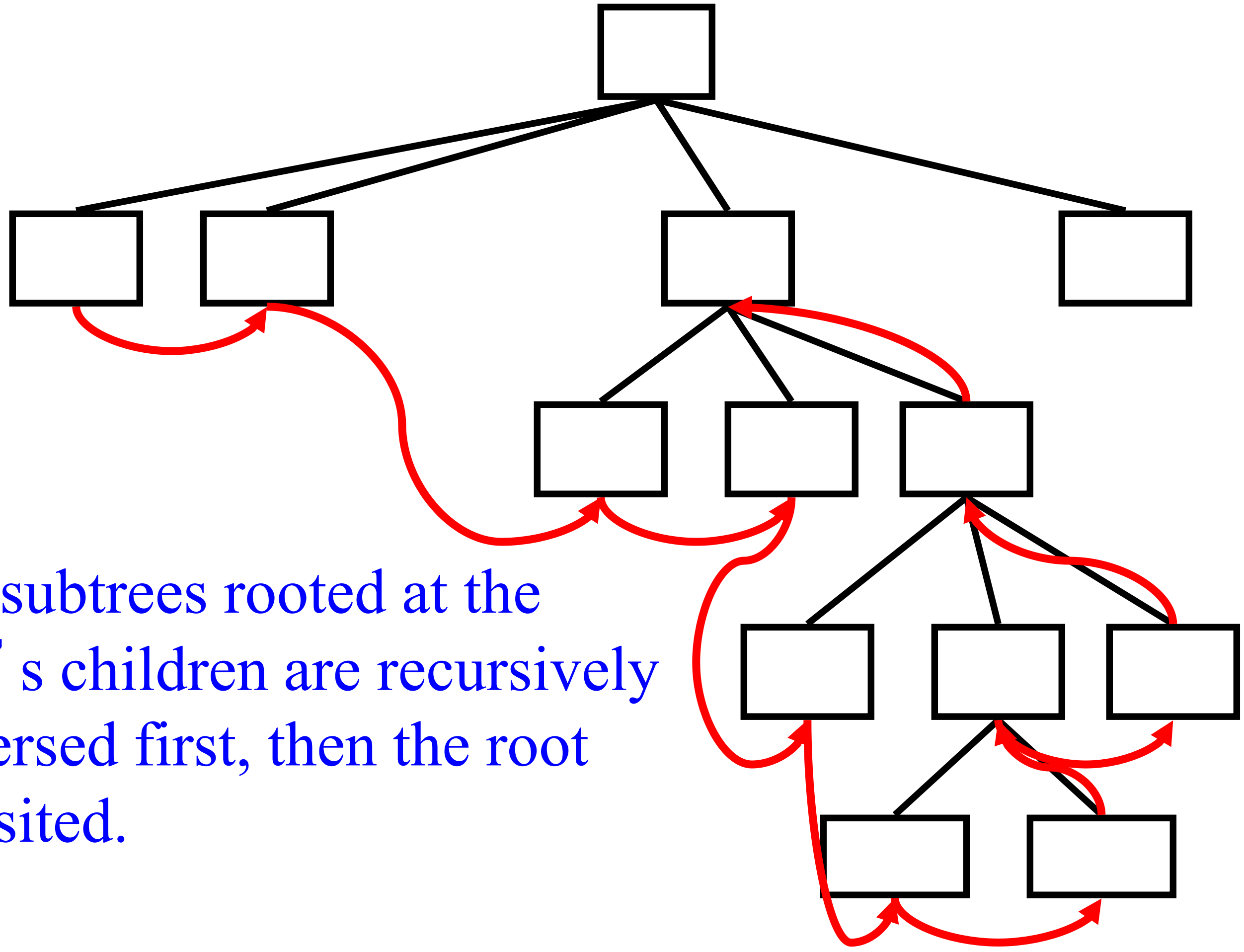
Postorder Traversal



The subtrees rooted at the root’s children are recursively traversed first, then the root is visited.

.

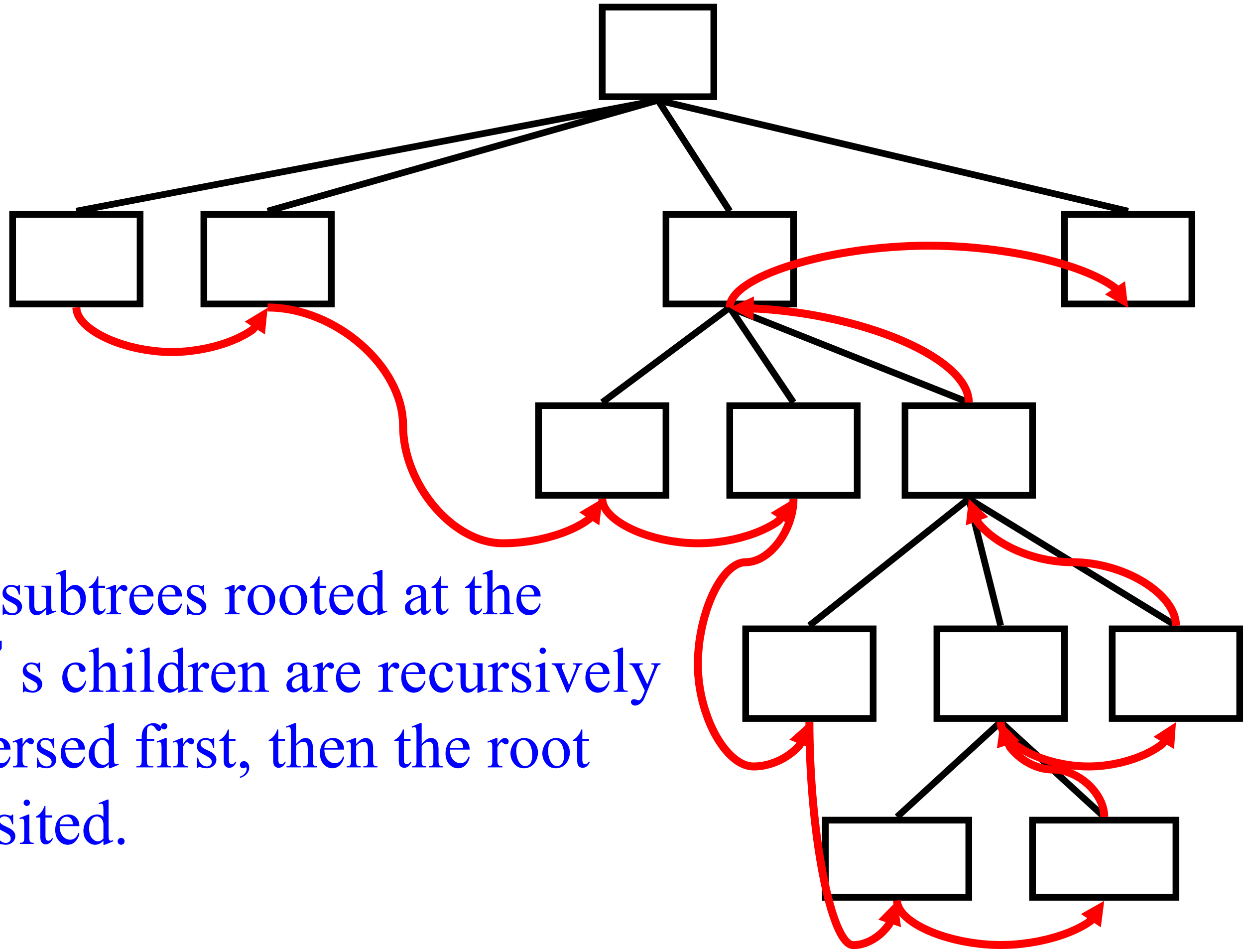
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

.

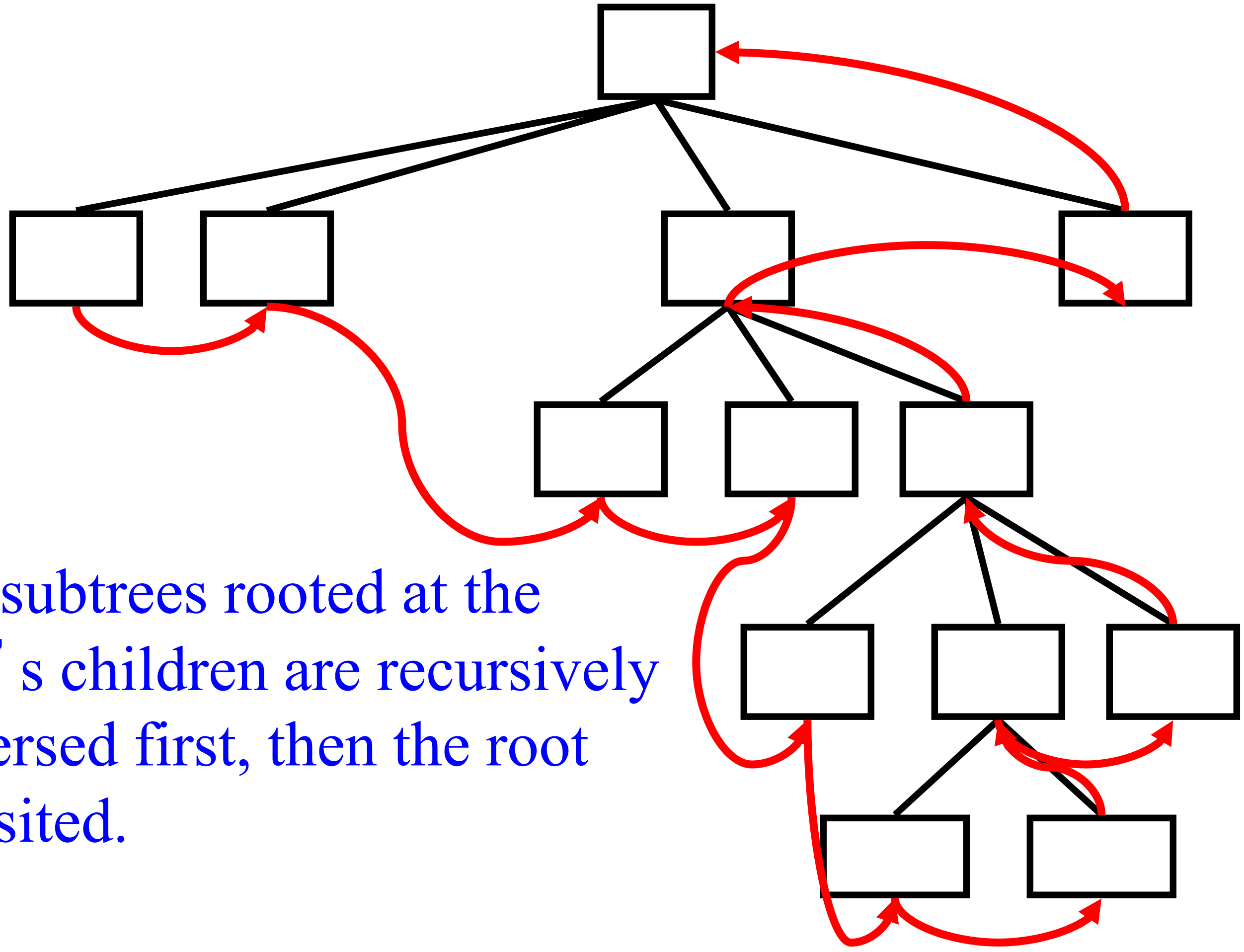
Postorder Traversal



The subtrees rooted at the root’s children are recursively traversed first, then the root is visited.

.

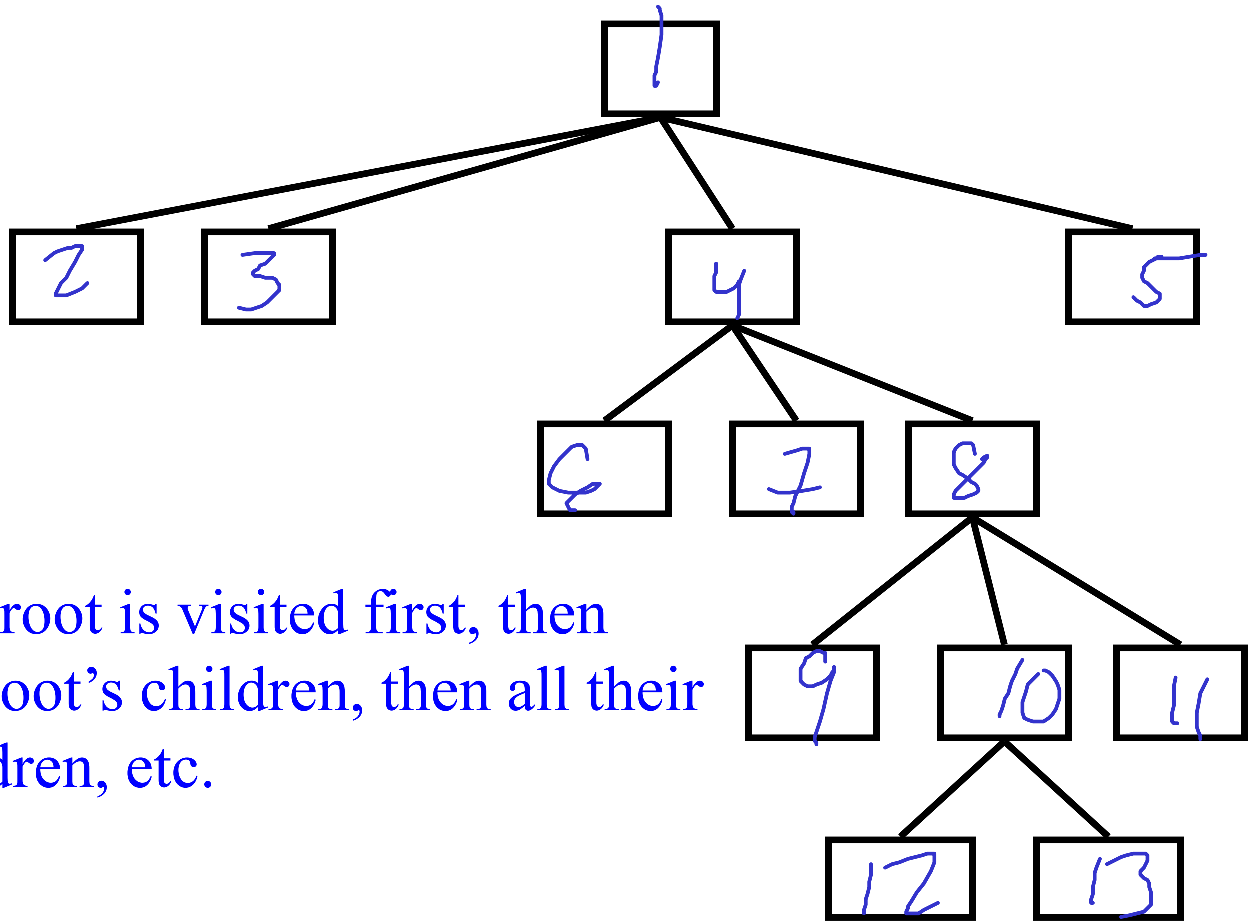
Postorder Traversal



The subtrees rooted at the root's children are recursively traversed first, then the root is visited.

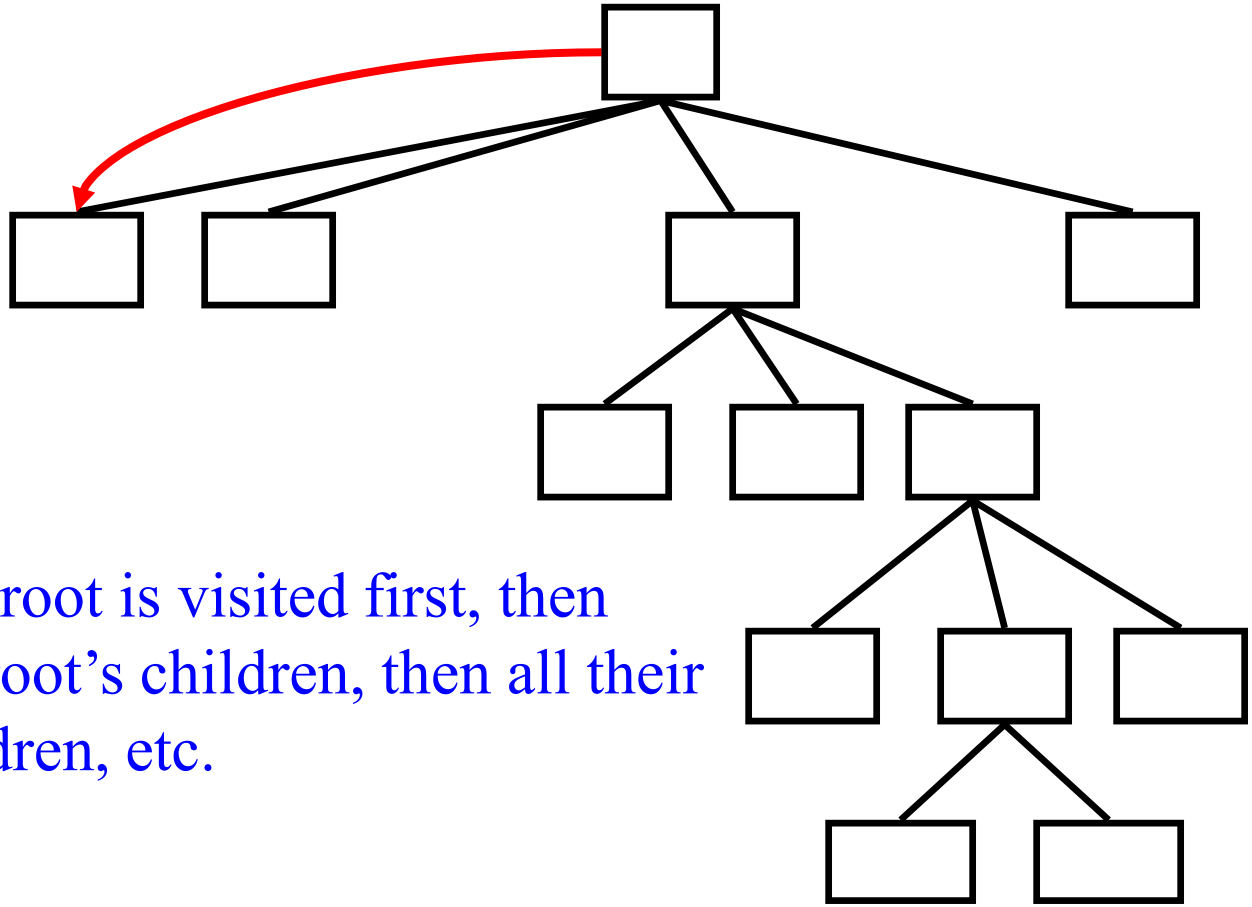
.

Level-order Traversal
Breadth First Search



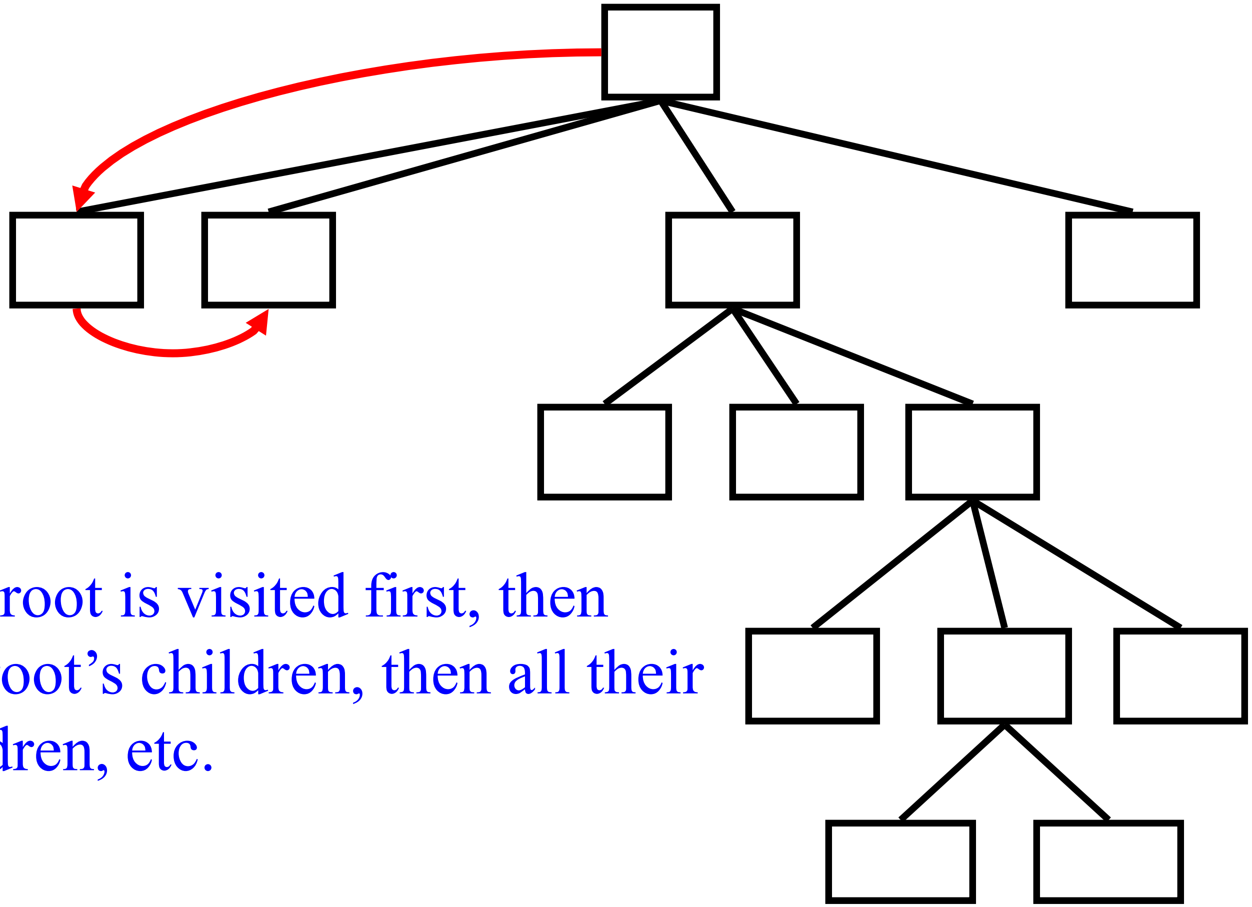
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



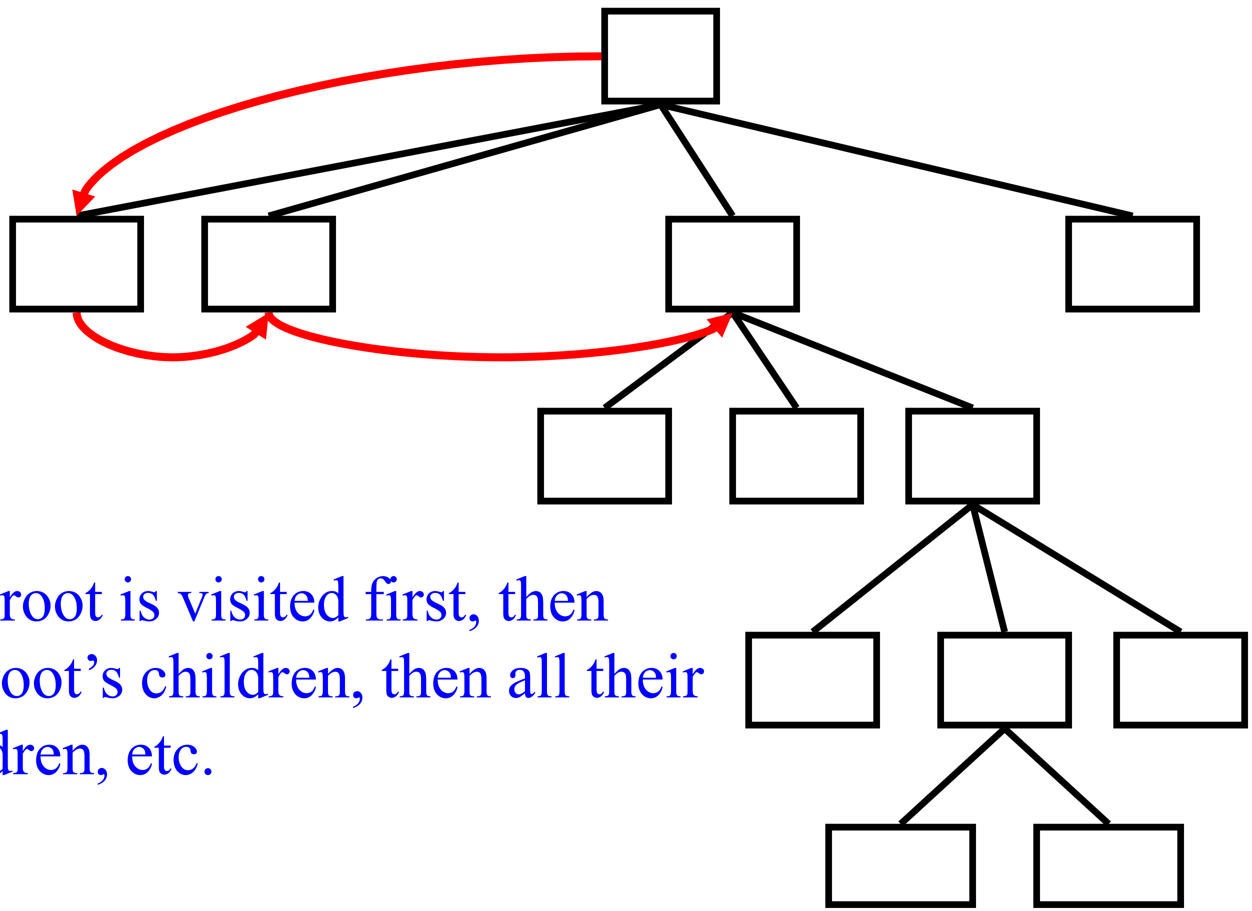
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



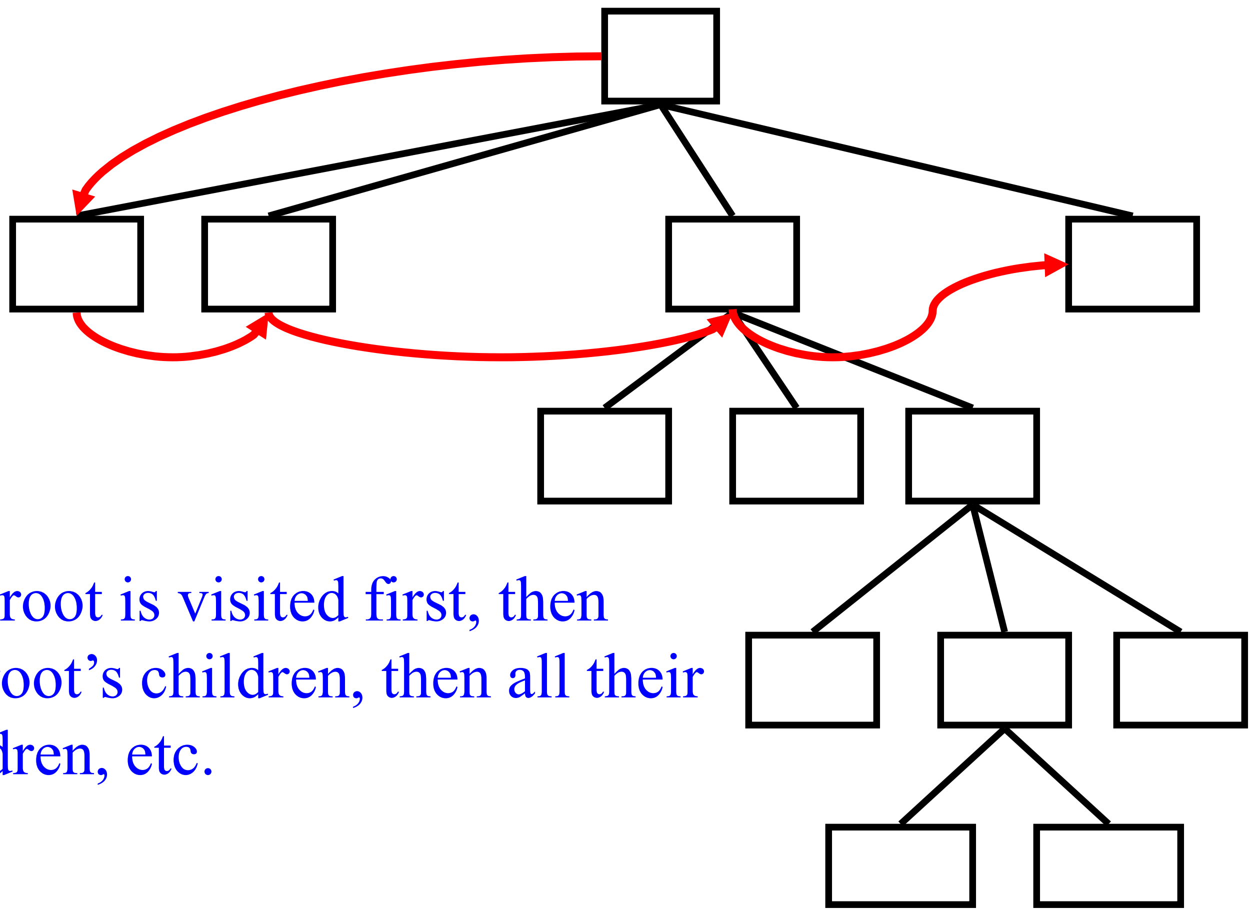
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



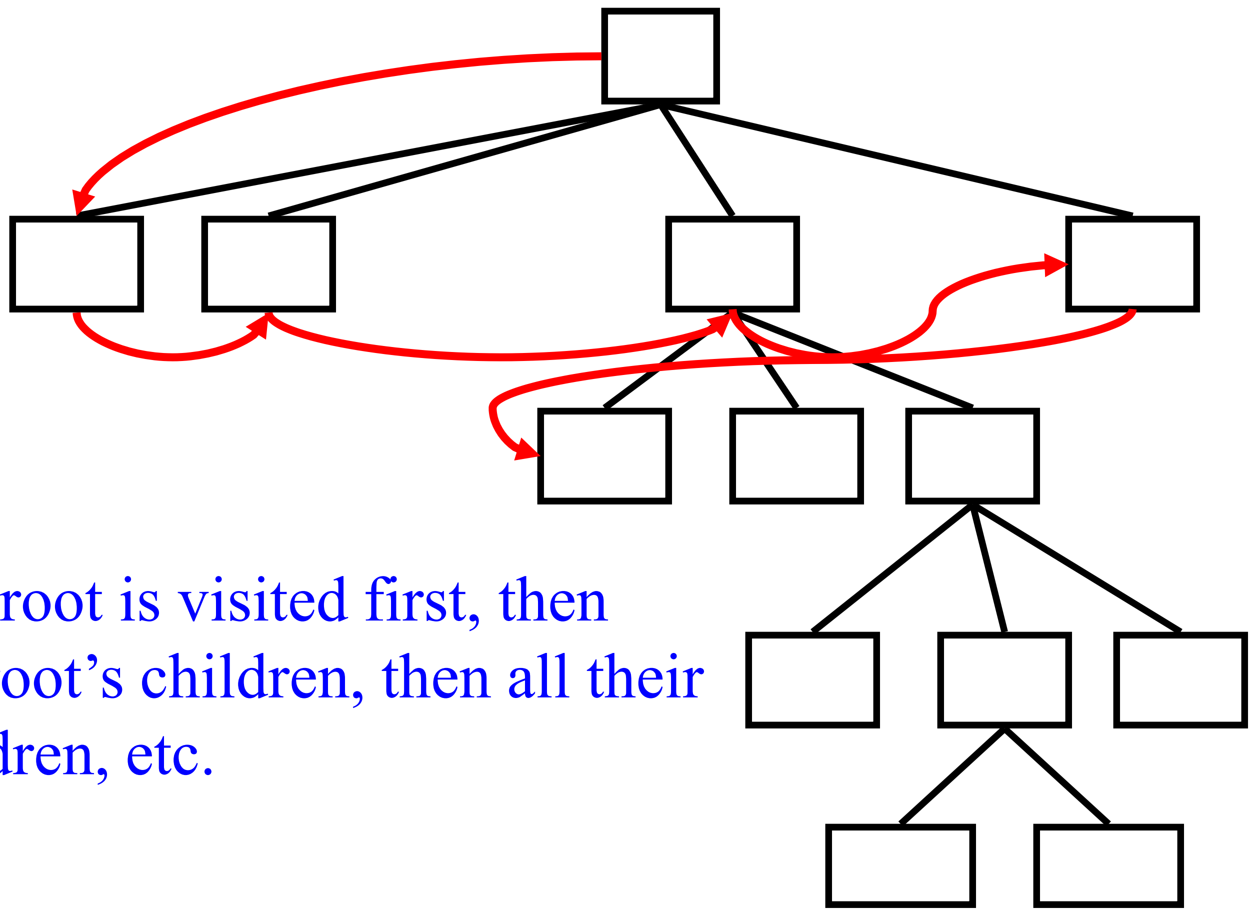
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



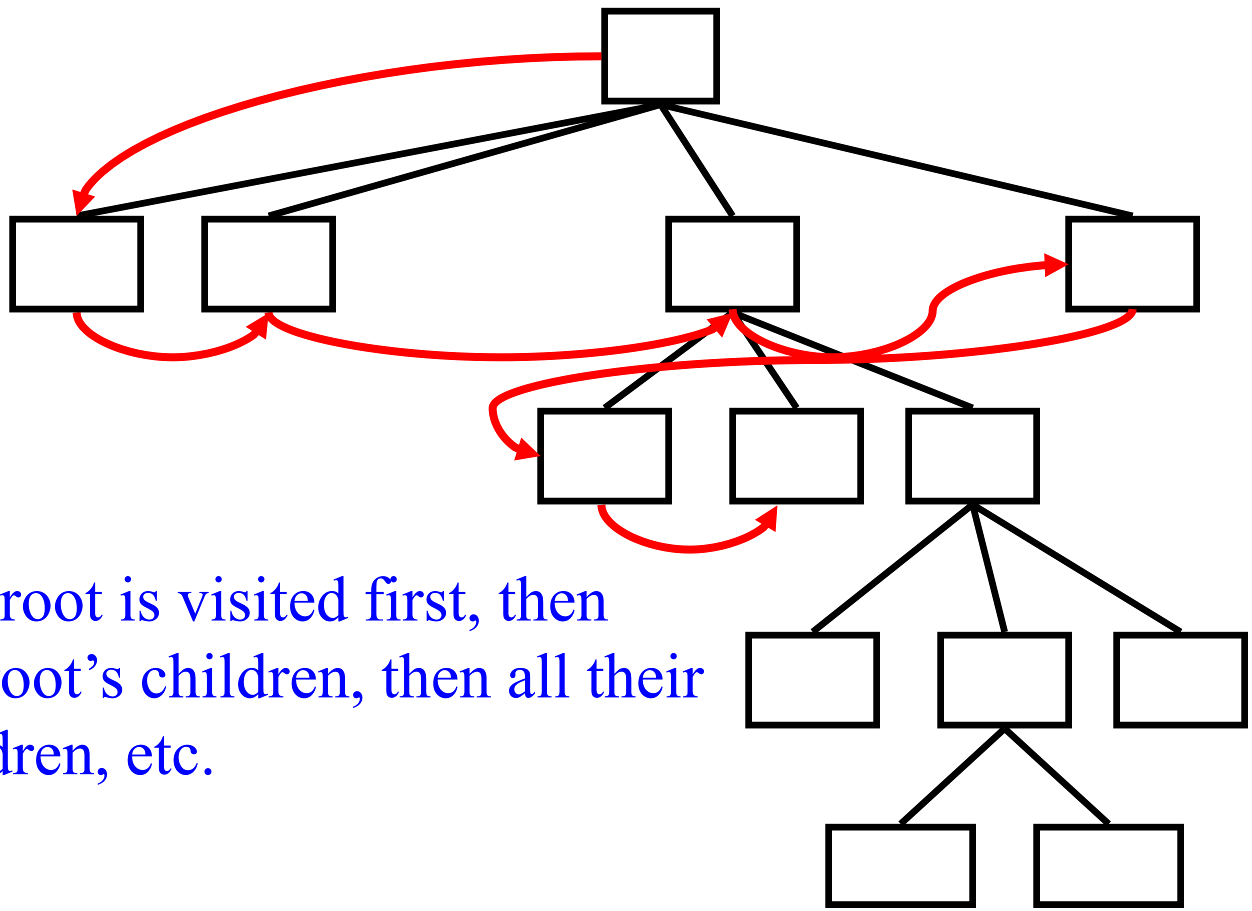
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



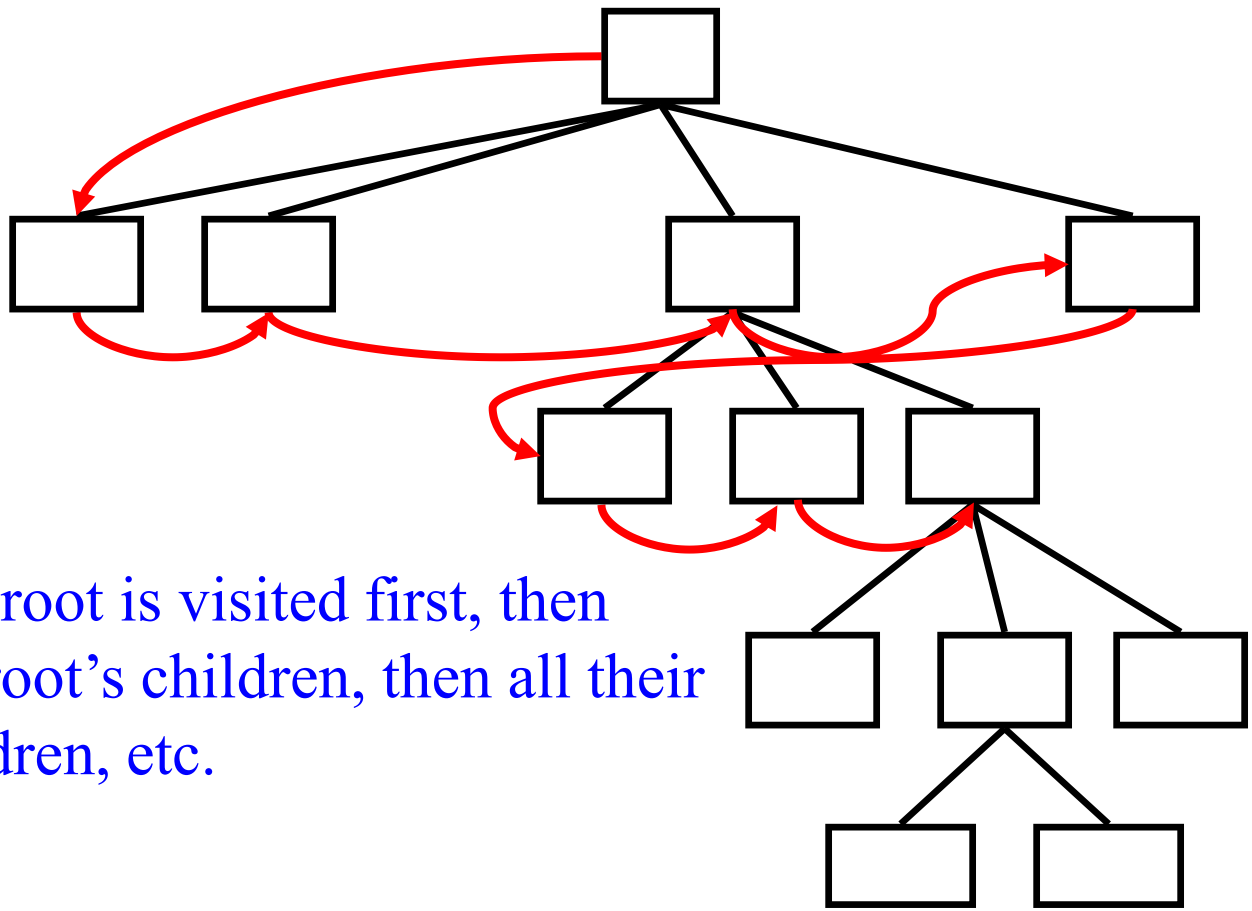
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



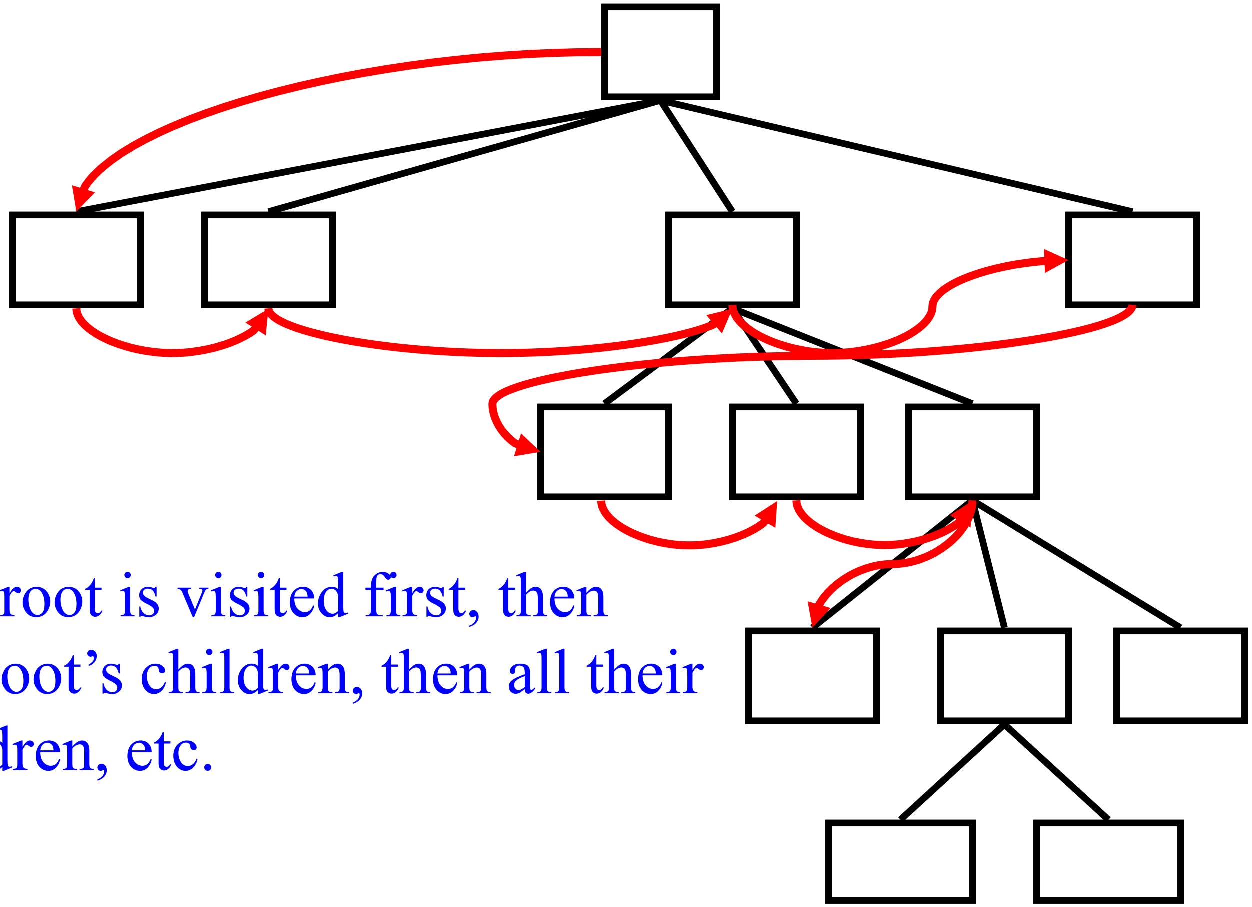
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



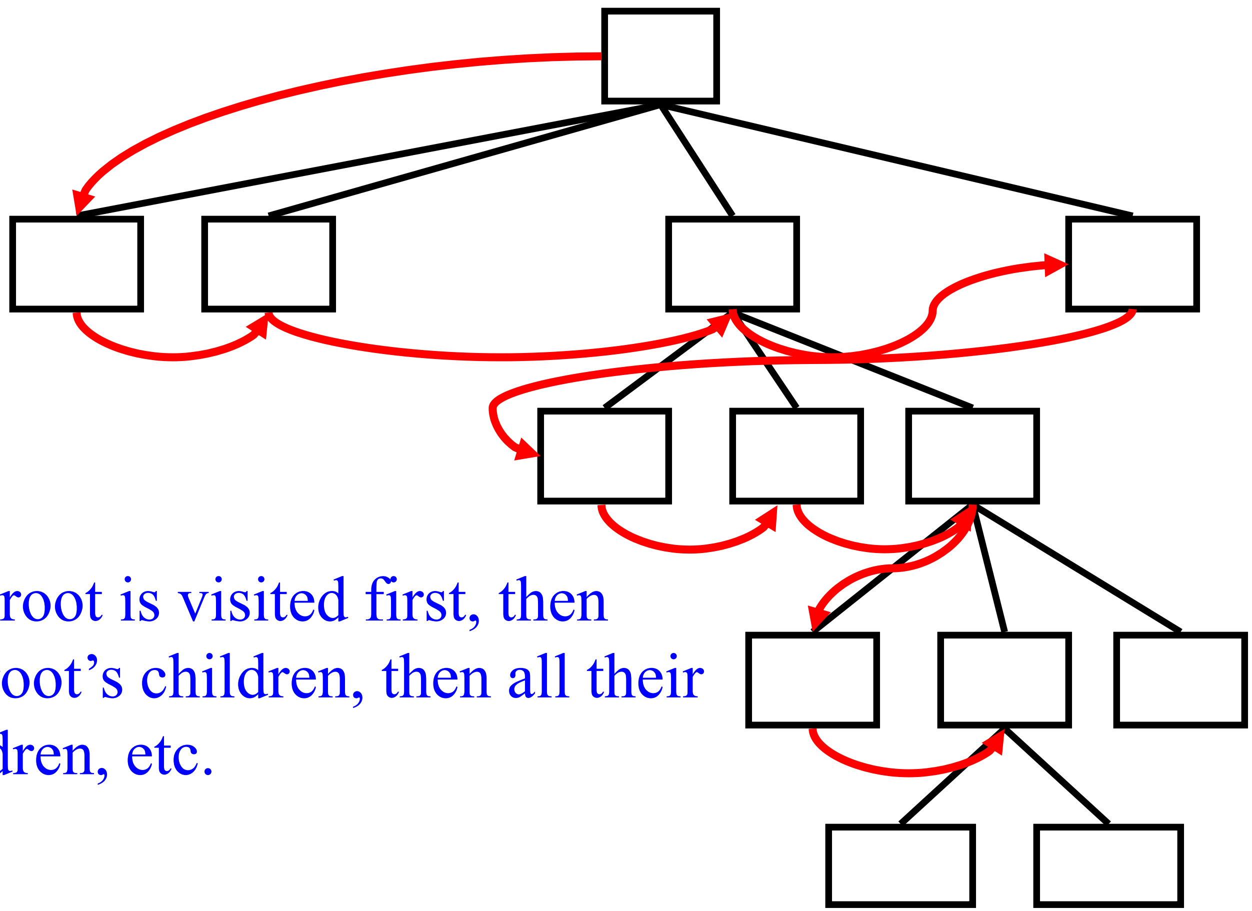
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



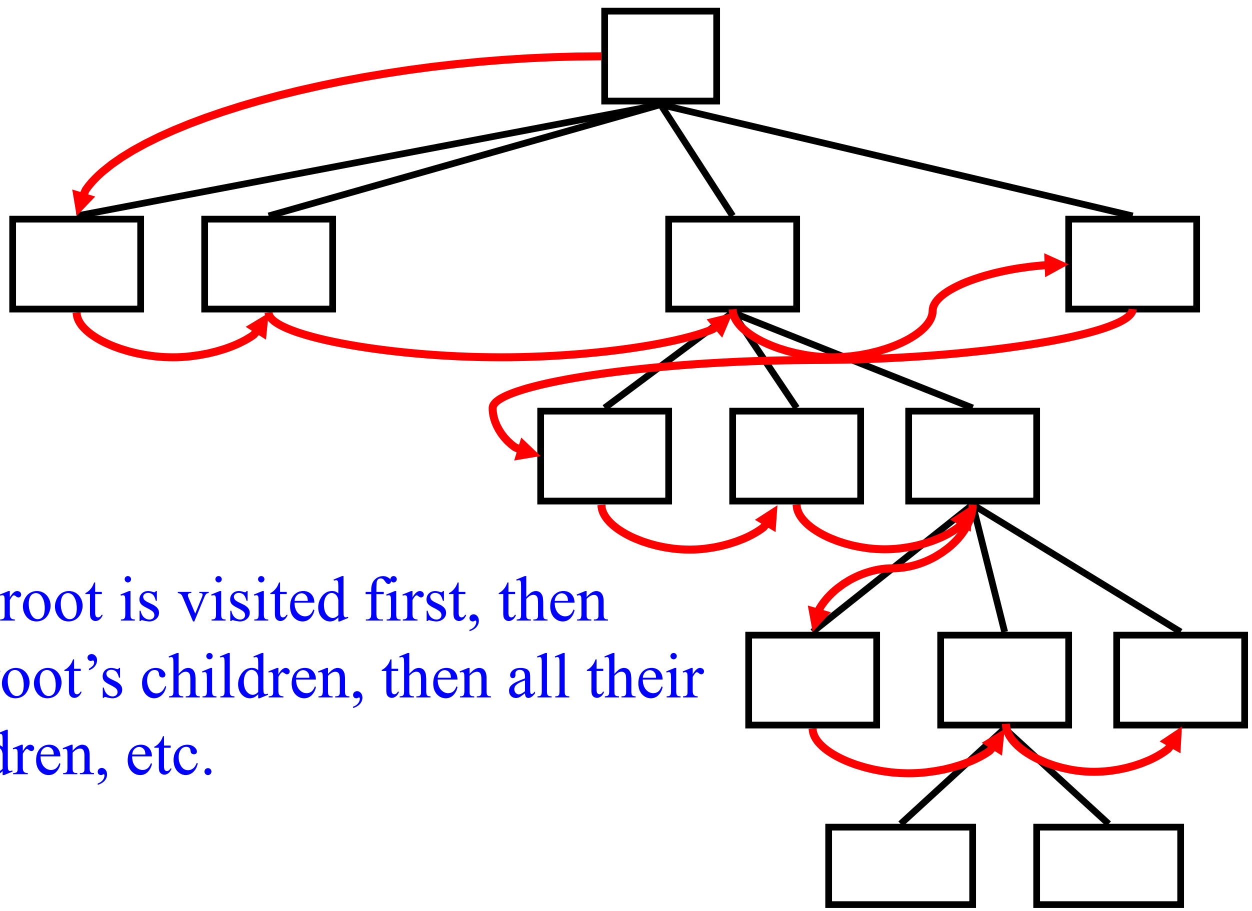
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



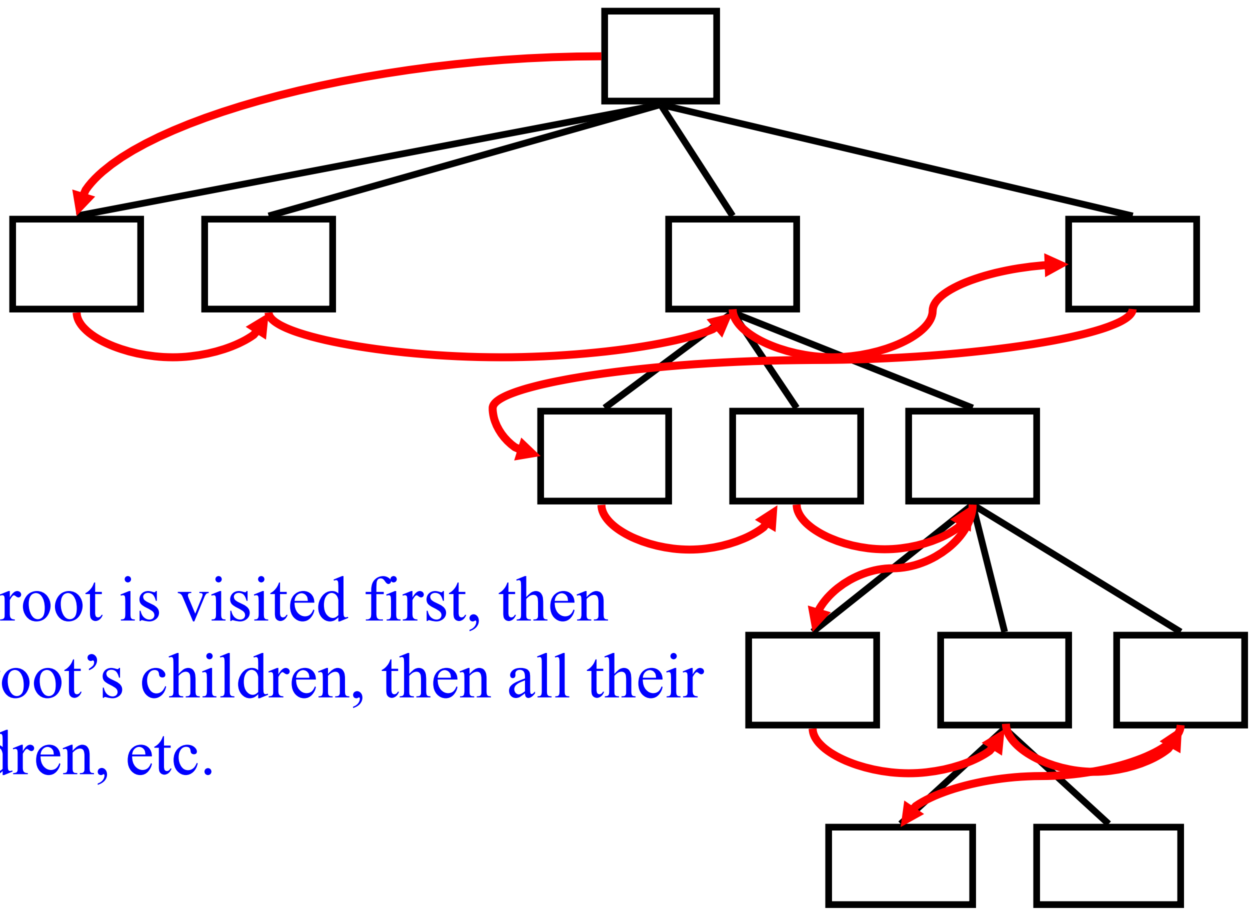
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



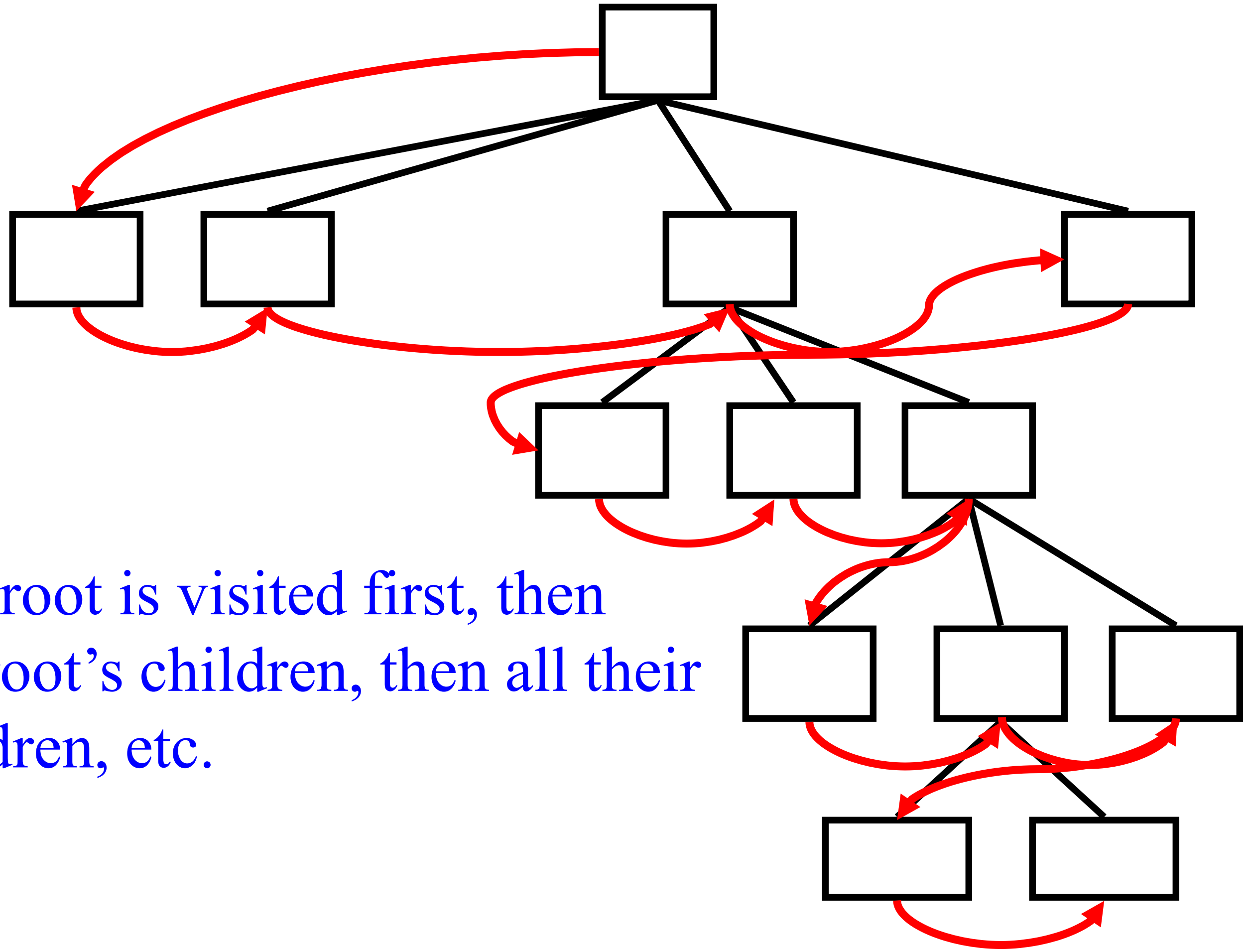
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



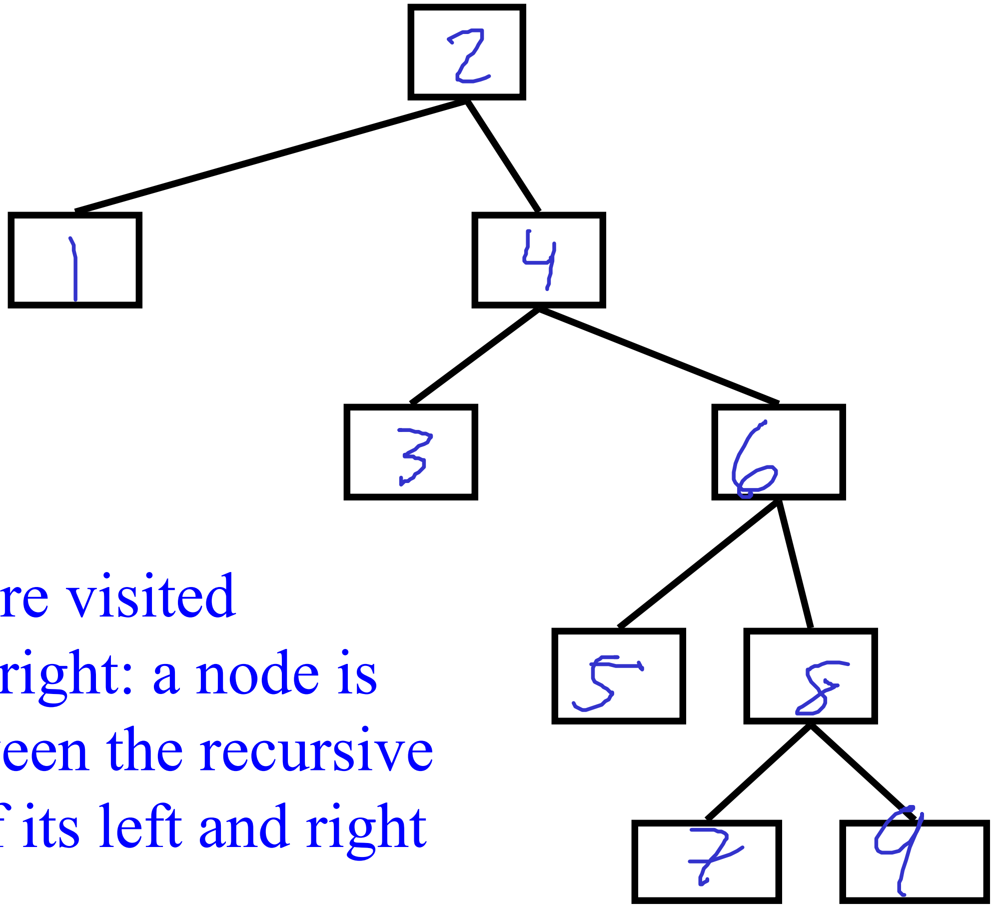
The root is visited first, then the root's children, then all their children, etc.

Level-order Traversal
Breadth First Search



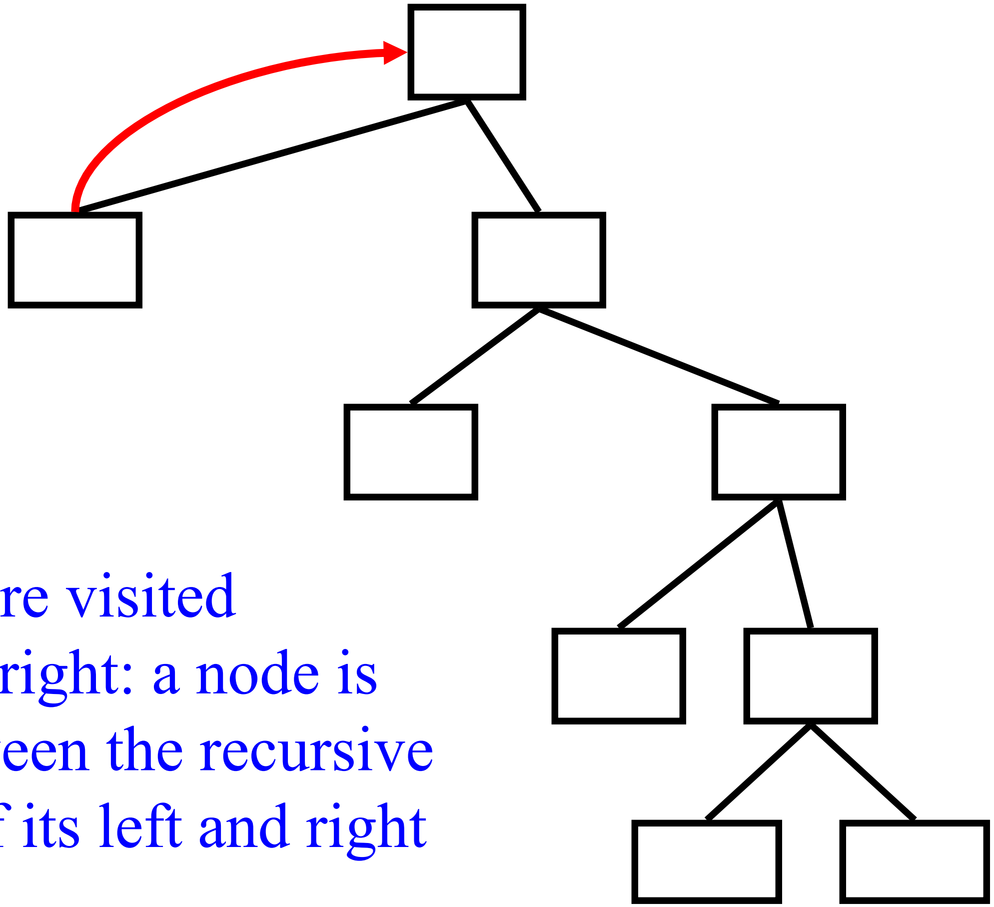
The root is visited first, then the root's children, then all their children, etc.

Inorder Traversal for Binary Trees



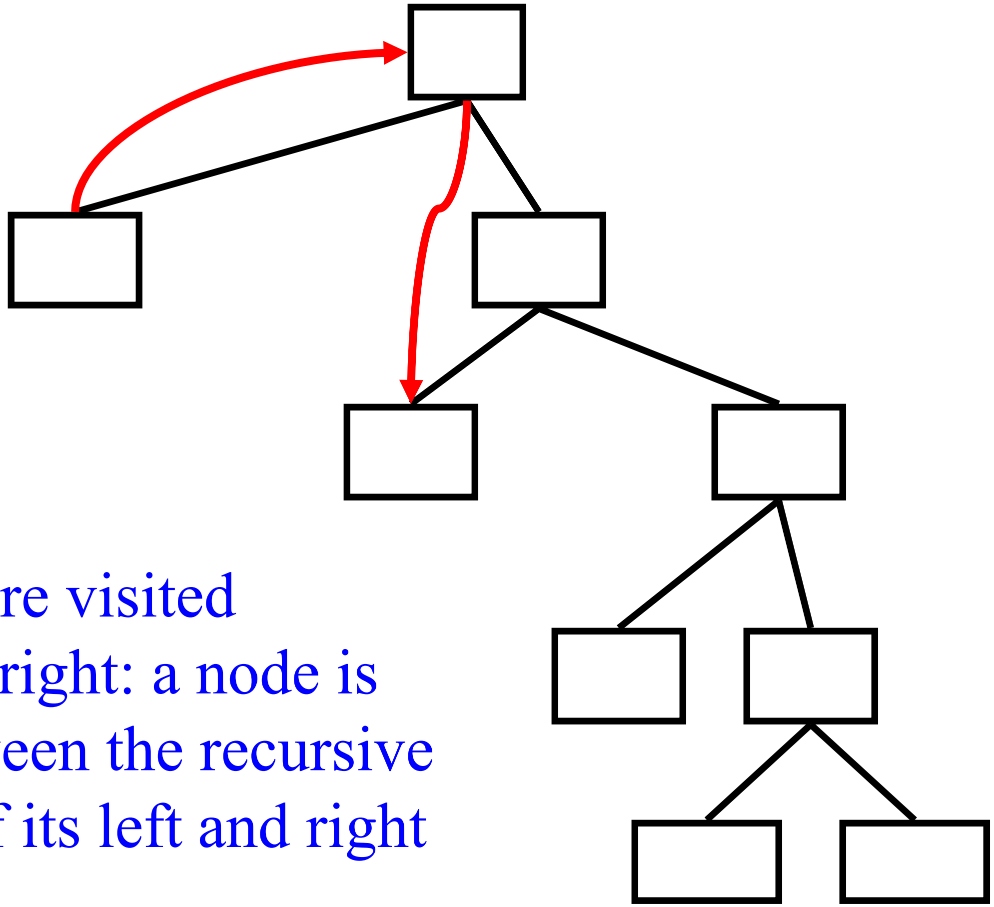
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



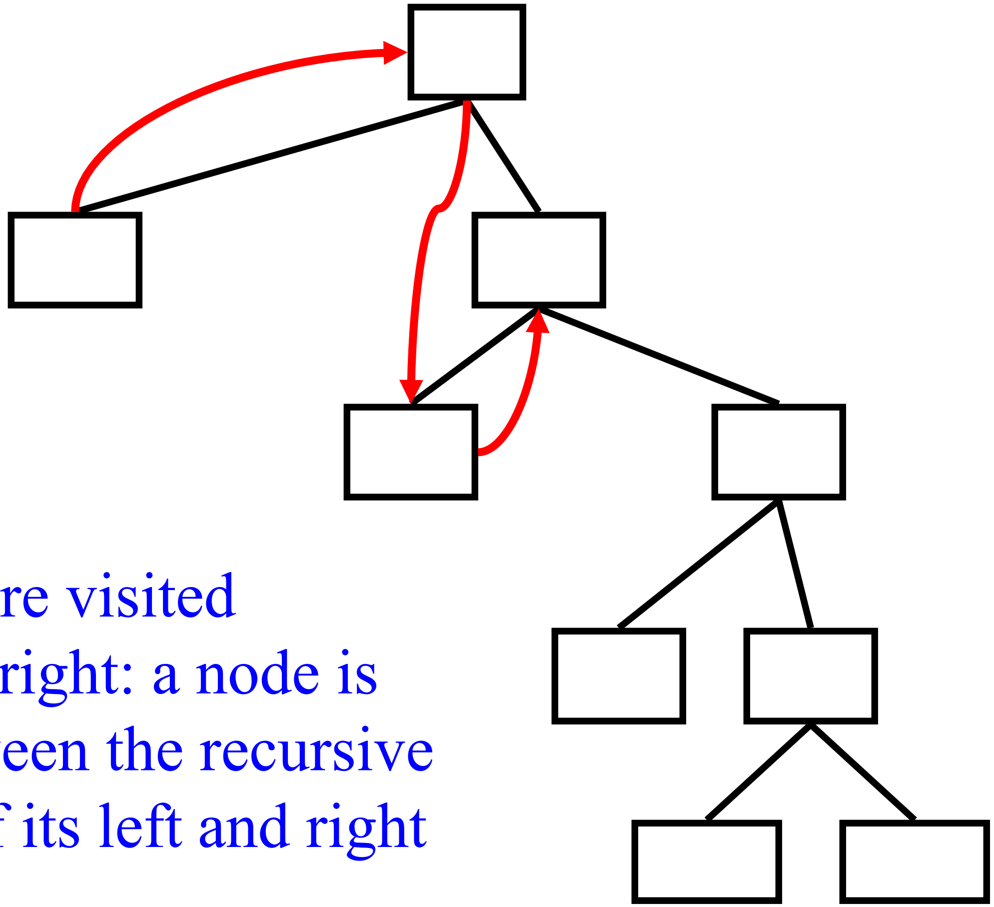
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



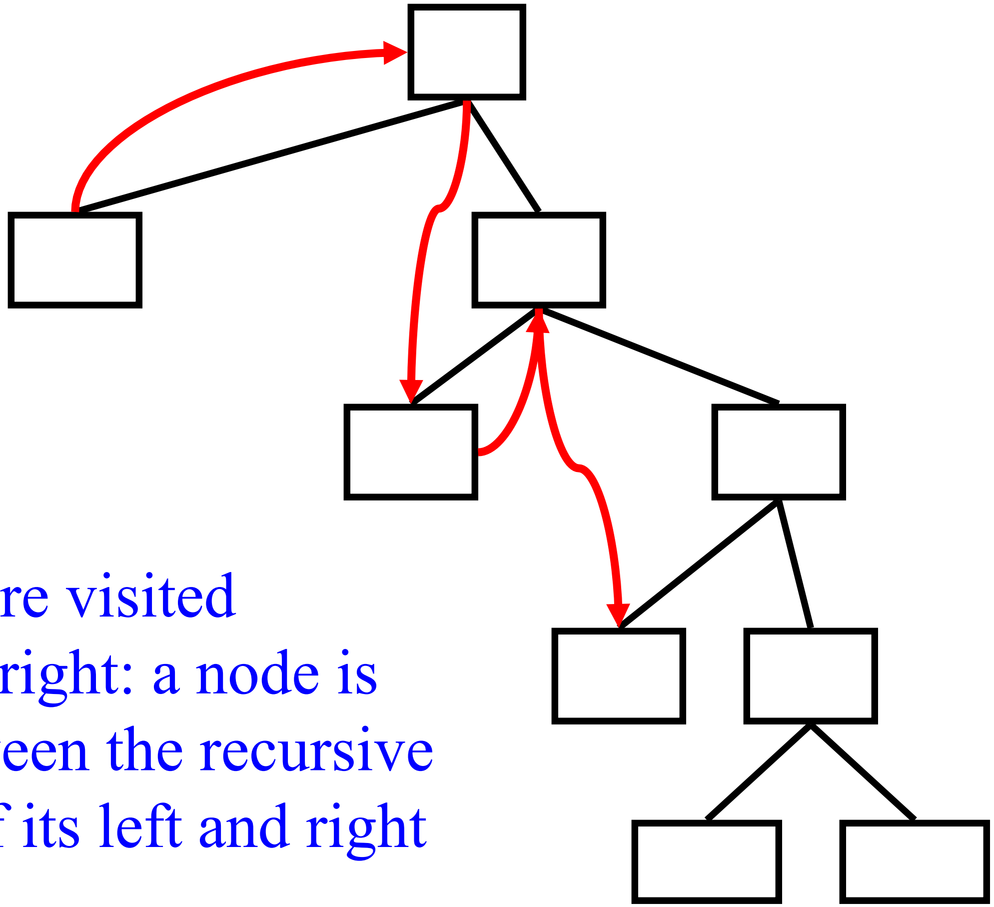
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



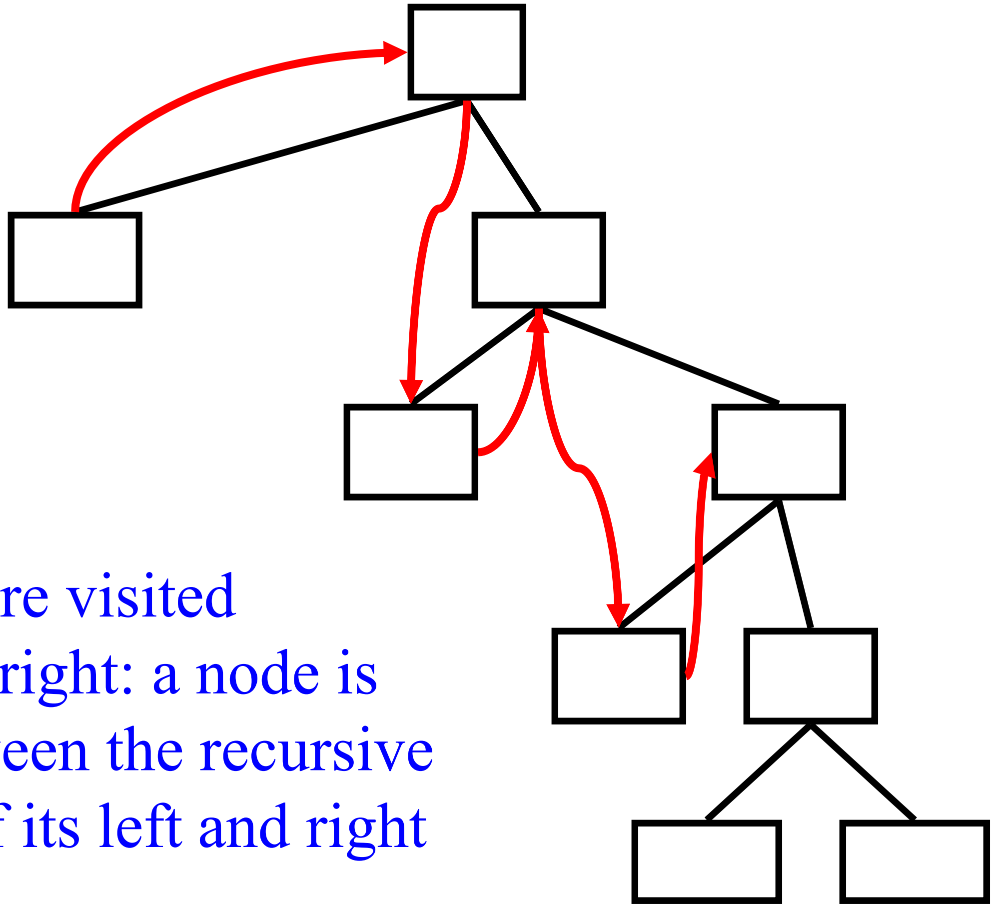
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



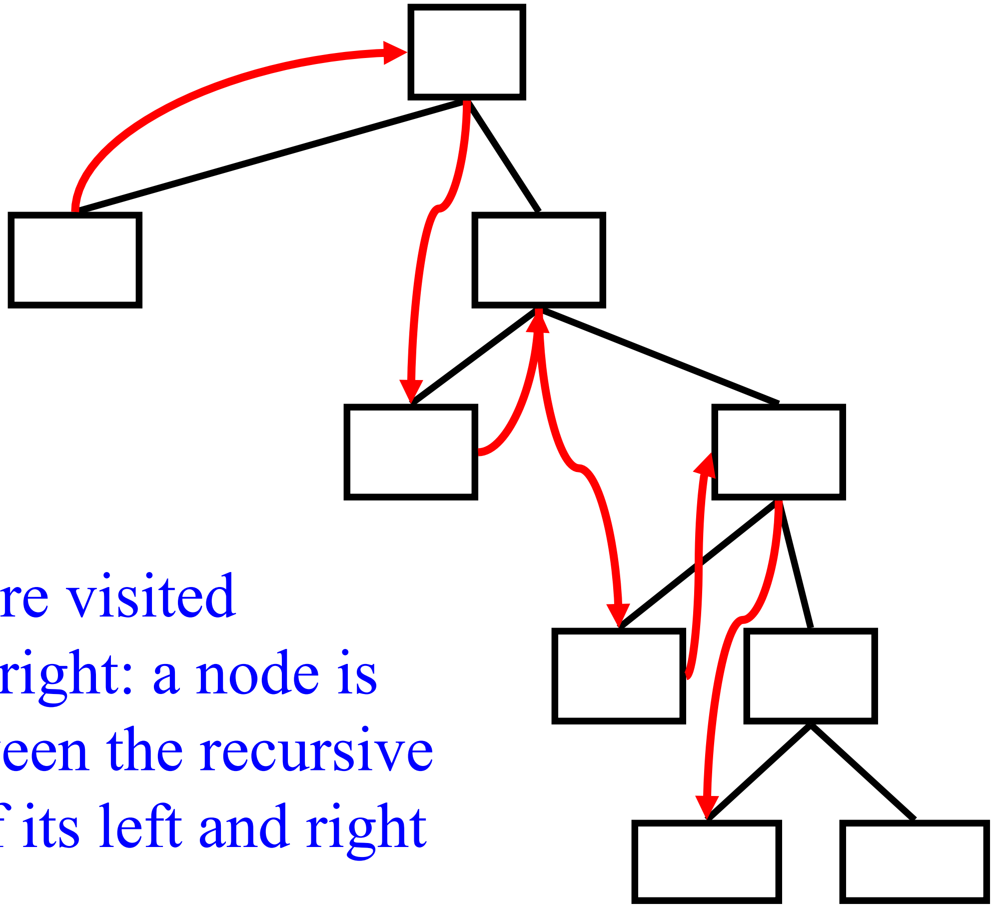
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



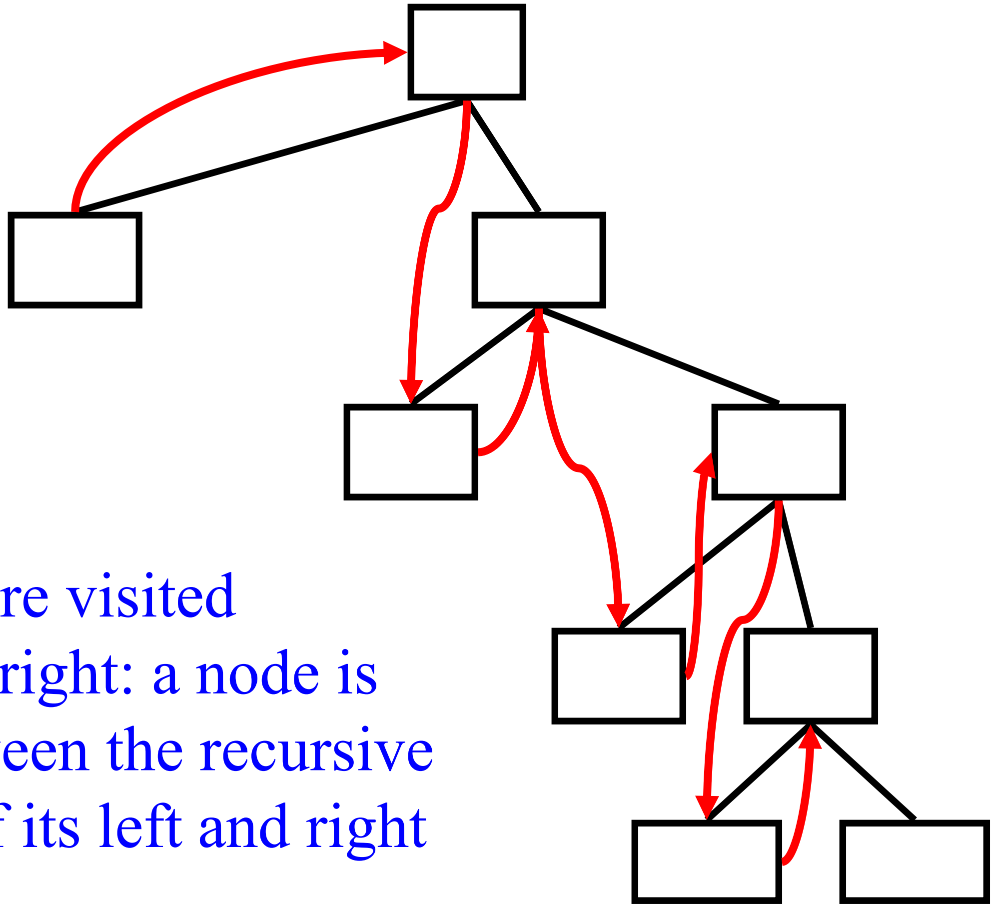
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



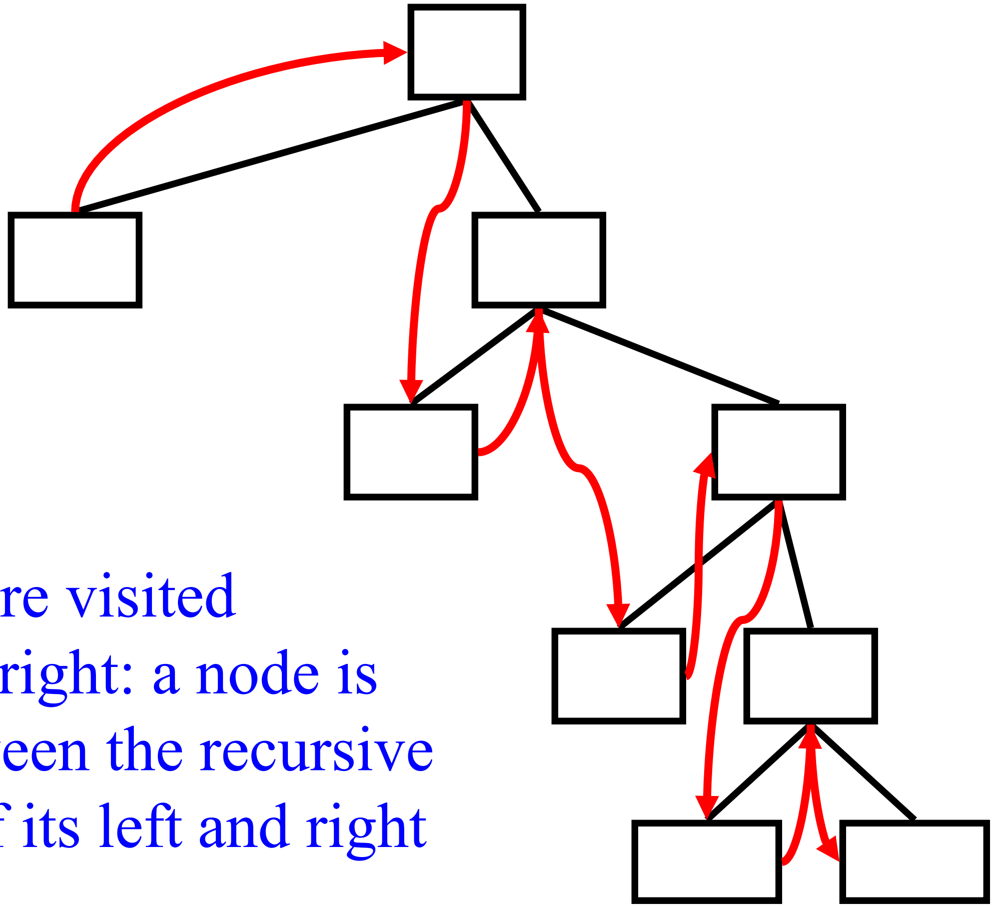
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



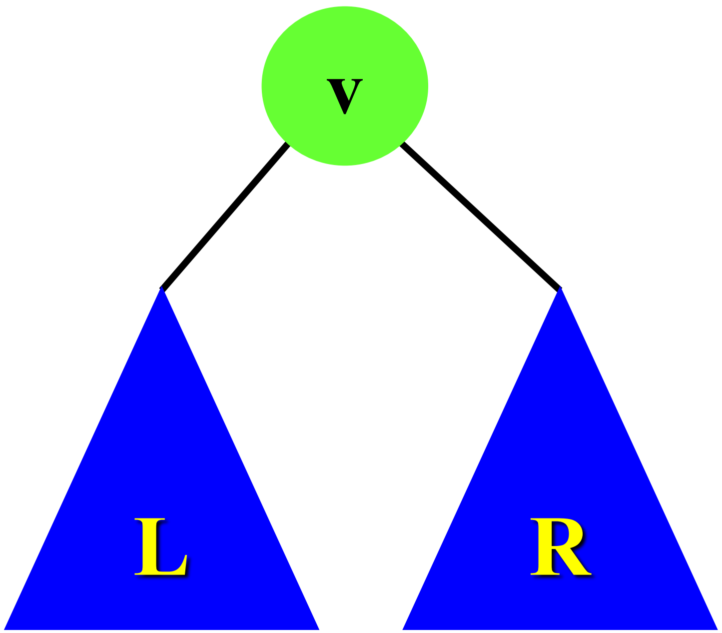
The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Inorder Traversal for Binary Trees



The nodes are visited from left to right: a node is visited between the recursive traversals of its left and right subtrees.

Binary Tree Traversals



- Preorder
  - v, L, R
- Postorder
  - L, R, v
- Inorder
  - L, v, R
- Levelorder
  - v, vL, vR, L, R

Preorder Traversal
Depth First Search (DFS)

Algorithm preorder(Tree T, Node v):
processNode(v)
if T.isInternal(v) **then**
preorder(T, T.leftChild(v))
preorder(T, T.rightChild(v))
end
end

Inorder Traversal

Algorithm inorder(Tree T, Node v):
    if T.isInternal(v) **then**
        inorder(T, T.leftChild(v))
    end
    processNode(v)
    if T.isInternal(v) **then**
        inorder(T, T.rightChild(v))
    end
end

Postorder Traversal

Algorithm postorder(Tree T, Node v):
  if T.isInternal(v) **then**
    postorder(T, T.leftChild(v))
    postorder(T, T.rightChild(v))
  end
  processNode(v)
end

Tree Traversal Numberings

pre = 1;   in = 1;   post = 1

algorithm eulerNumberings(Tree *T*, Node *v*)

  if *v* != null **then**

    v.pre = pre;   pre = pre + 1

    eulerNumberings(*T*,*T*.leftChild(*v*))

    v.in = in;   in = in + 1

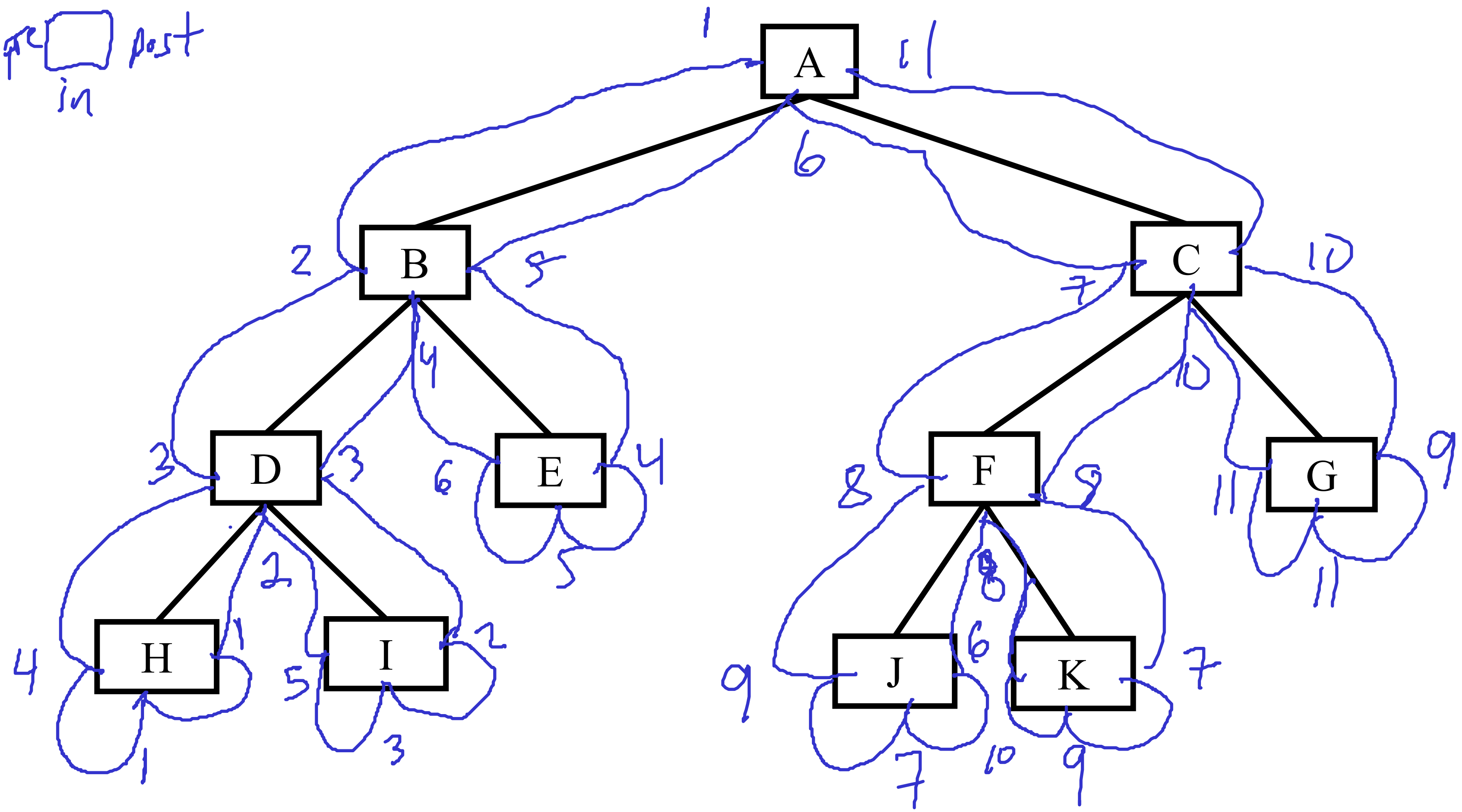
    eulerNumberings(*T*,*T*.rightChild(*v*))

    v.post = post;   post = post + 1

  end

end

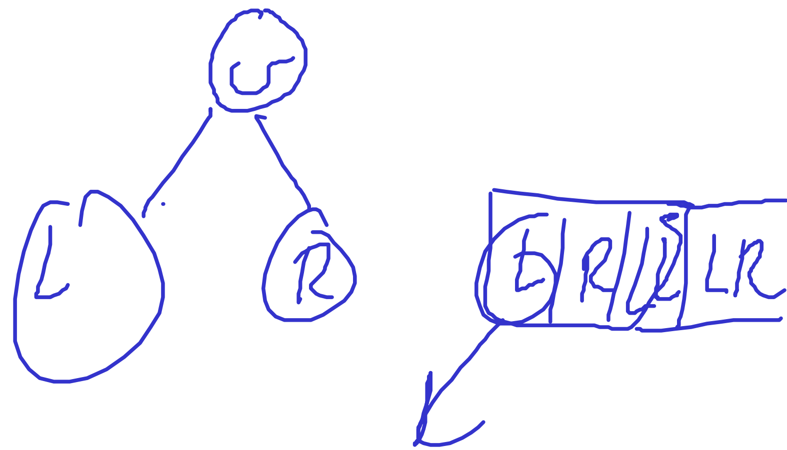
Tree Traversal Numbering Example



Levelorder Traversal
Breadth First Search (BFS)

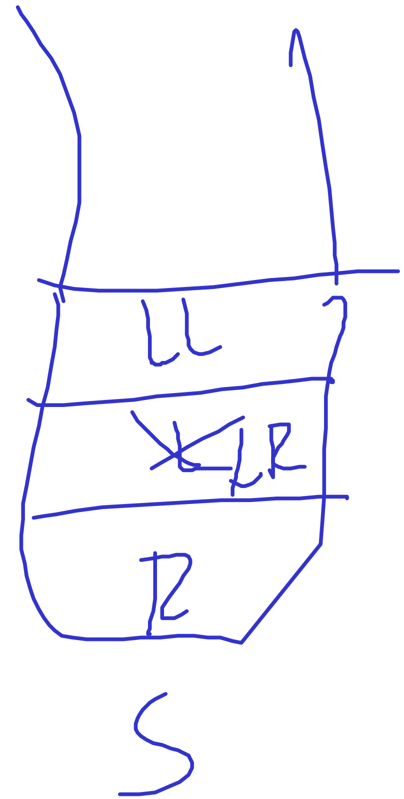
Algorithm levelorder(Tree T, Node r):

```
Algorithm levelorder(Tree T, Node r):
    Queue Q
    Q.enqueue(T.r)
    while !Q.empty() do
        v ← Q.dequeue()
        processNode(v)
        if T.isInternal(v) then
                Q.enqueue(T.leftChild(v))
                Q.enqueue(T.rightChild(v))
        end
    end
end
```



Non-recursive Preorder Traversal
Depth First Search (DFS)

Algorithm dfs(Tree T, Node r)
    Stack S
    S.push(r)
    while !S.empty() **do**
        $v \leftarrow S$.pop()
        processNode(v)
        if T.isInternal(v) **then**
                S.push(T.rightChild(v))
                S.push(T.leftChild(v))
        end
    end
end



Running Time of Tree Traversals

- Each node is visited a fixed number of times (i.e., 3 times)
- **Theorem**
  - The time complexity of Preorder, Inorder, Postorder is $O(n)$

$T(n) = T(n-1) + O(1)$

$\in \underline{O(n)}$

Tree Traversal Summary

Preorder	Depth-first	Stack
Inorder	Symmetric	Stack
Postorder	Bottom-up	Stack
Levelorder	Breadth-first	Queue

Data Structures for *binary* Trees

Operation	Time with linked structure
positions, elements, traversals (iterators): pre-, in-, post-, level-order	$O(n)$
size, isEmpty	O(1)
swapElements, replaceElement	O(1)
leftChild, rightChild, sibling	O(1)
isInternal, isExternal, isRoot	O(1)
root, parent, child	O(1)