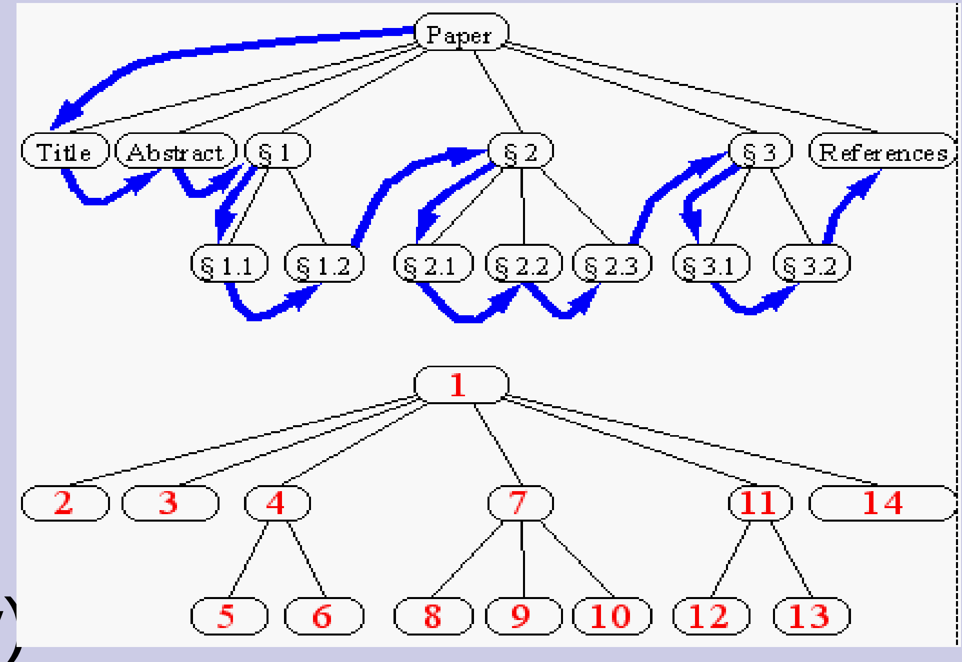
Tree Walks/Traversals

- A tree walk or traversal is a way of visiting all the nodes in a tree in a specified order.
- A preorder tree walk processes each node before processing its children
- A postorder tree walk processes each node after processing its children

Traversing Trees (preorder)



- preorder traversal

  Algorithm preOrder(v)

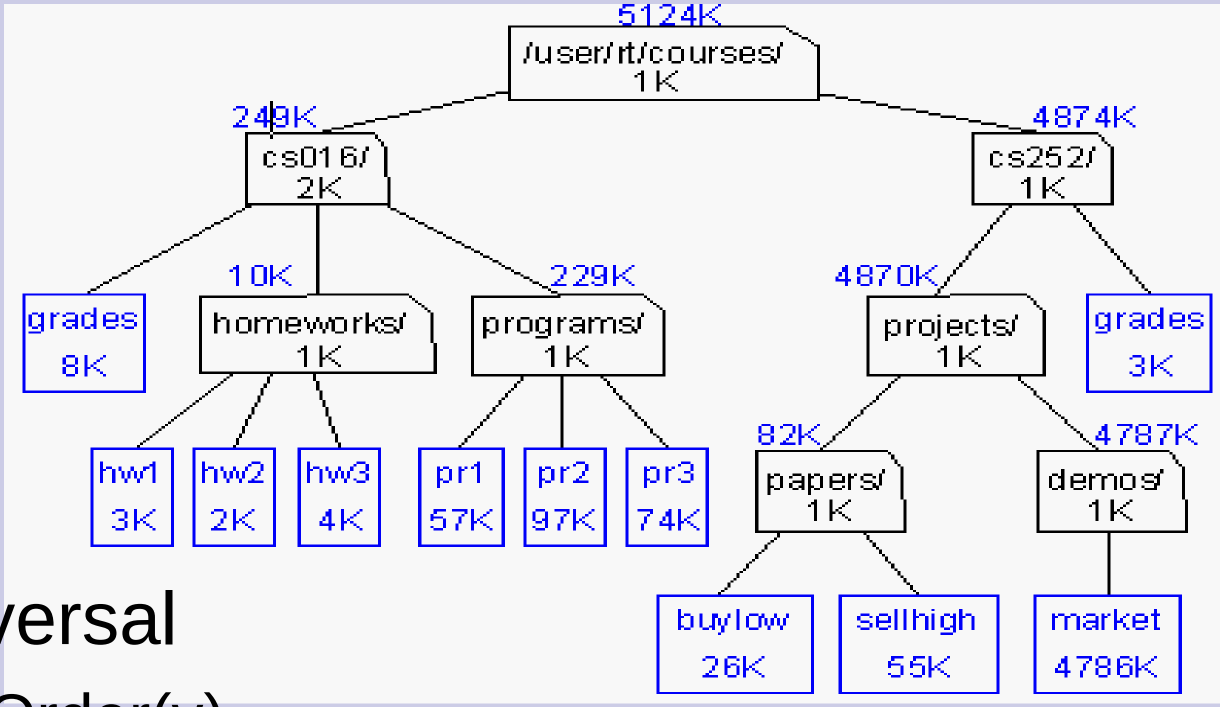
  "visit" node v

  for each child w of v do

  recursively perform preOrder(w)

- reading a document from beginning to end

Traversing Trees (postorder)



- postorder traversal

  Algorithm postOrder(v)

  for each child w of v do

  recursively perform postOrder(w)

  “visit” node v

- du (disk usage) command in Unix

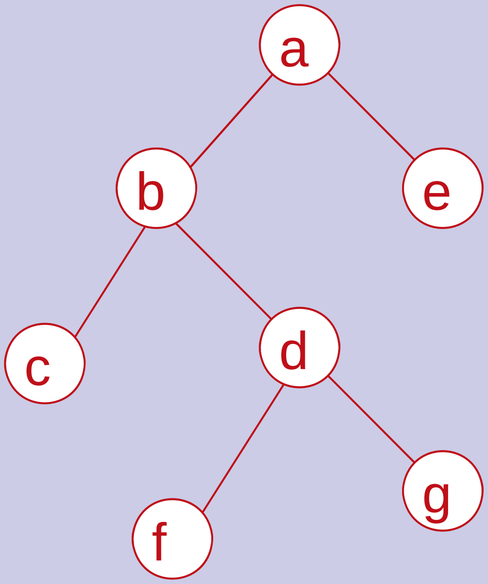
Traversals of Binary Trees

```
preorder(v)
   if (v == null) then return
   else  visit(v)
            preorder(v.leftchild())
            preorder(v.rightchild())

postorder(v)
   if (v == null) then return
   else postorder(v.leftchild())
            postorder(v.rightchild())
            visit(v)
```

Examples of pre and postorder

- We assume that we are only printing the data in a node when we visit it.



Preorder

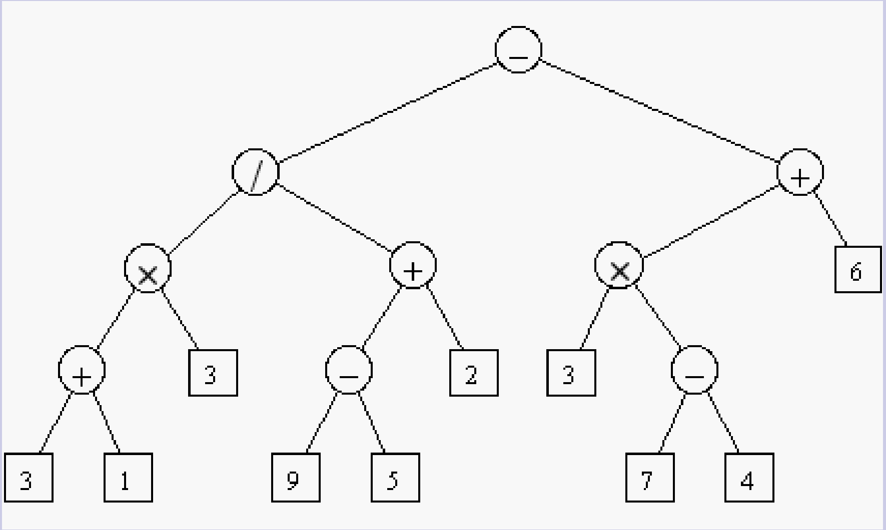
a b c d f g e

Postorder

c f g d b e a

Evaluating Arithmetic Expressions

- specialization of a postorder traversal



Algorithm evaluate(v)

    if v is a leaf

        return the variable stored at v

    else

        let o be the operator stored at v

        x $\rightarrow$ evaluate(v.leftChild())

        y $\rightarrow$ evaluate(v.rightChild())

        return x o y

Traversing Trees

- Besides preorder and postorder, a third possibility arises when v is visted between the visit to the left ad right subtree.

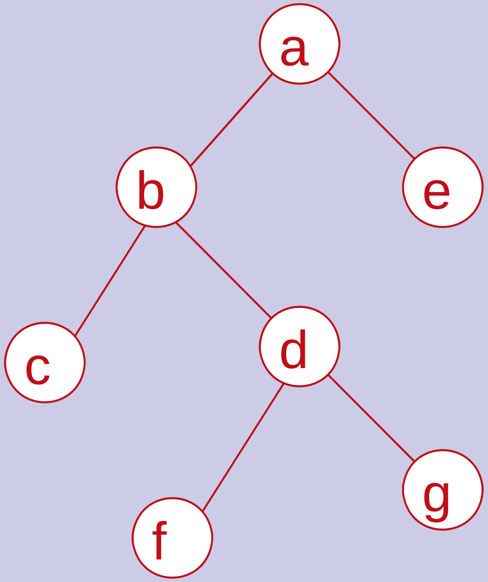
- **Algorithm** inOrder(v)

```
if (v == null) then return
else inOrder(v.leftChild())
        visit(v)
        inOrder(v.rightChild())
```

Inorder Example

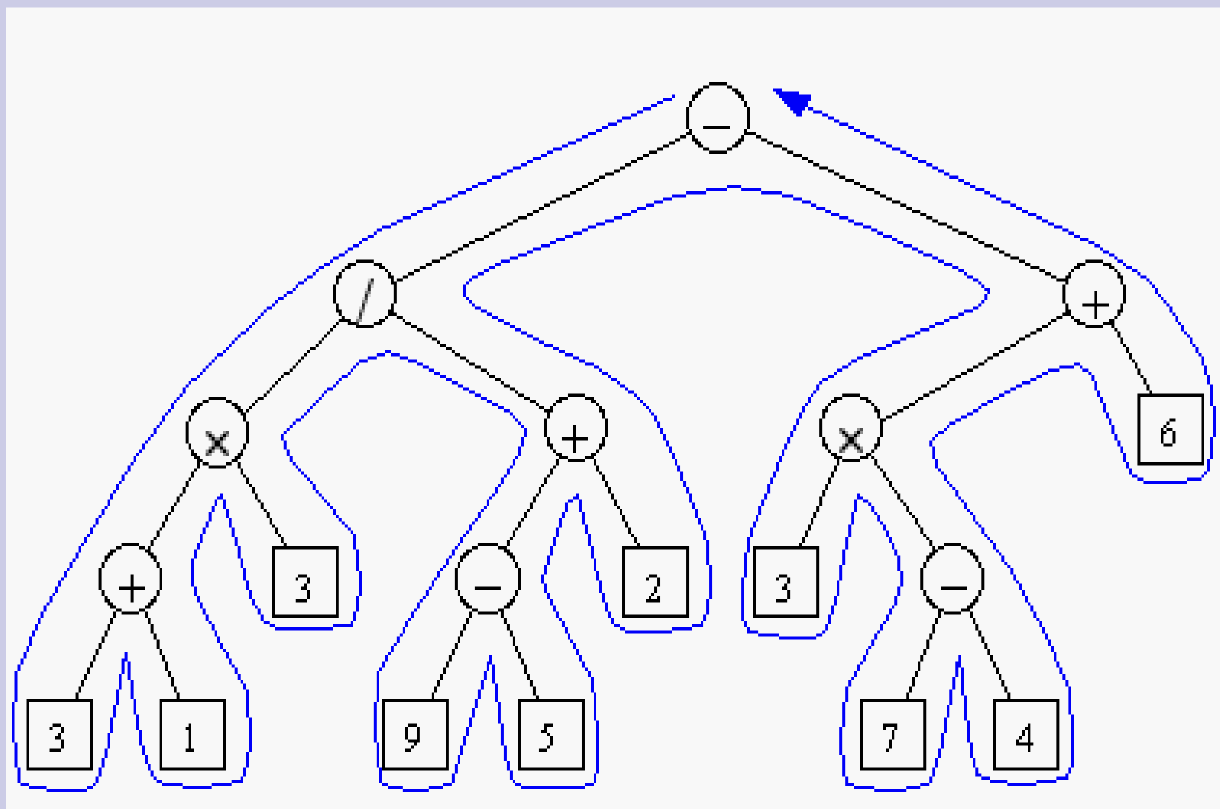
- Inorder

  c b f d g a e



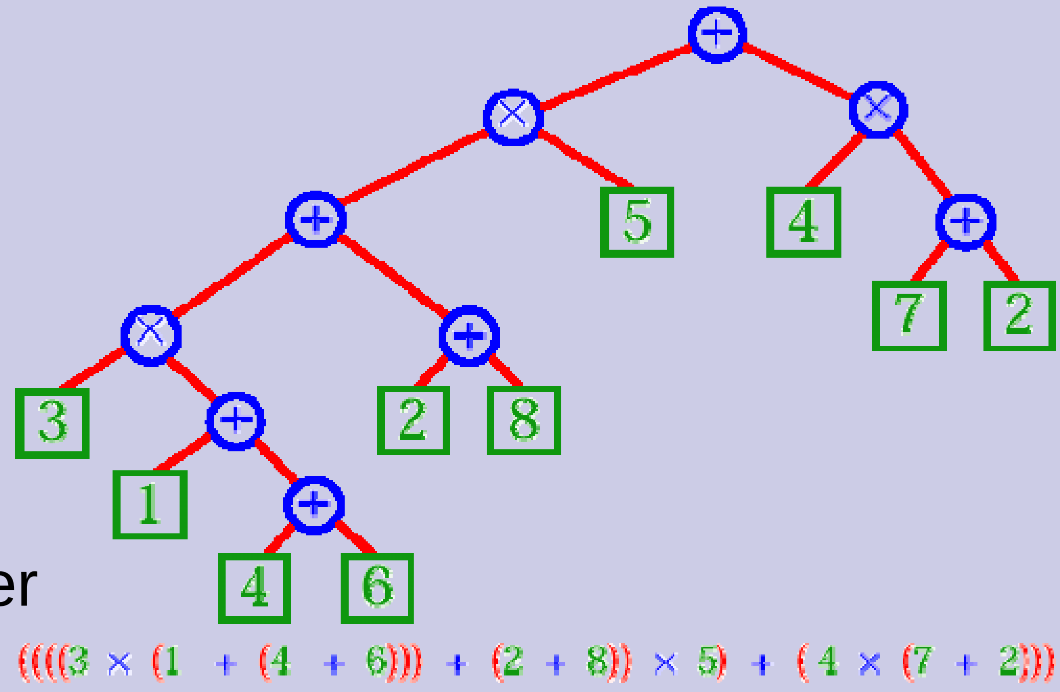
Euler Tour Traversal

- generic traversal of a binary tree
- the preorder, inorder, and postorder traversals are special cases of the Euler tour traversal
- “walk around” the tree and visit each node three times:
  - on the left
  - from below
  - on the right



Printing an arithmetic expression

- Printing an arithmetic expression - so called Euler’s walk:
  - Print “(“ before traversing the left subtree, traverse it
  - Print the value of a node
  - Traverse the right subtree, print “)” after traversing it



Template Method Pattern

- generic computation mechanism that can be specialized by redefining certain steps
- implemented by means of an abstract Java class with methods that can be redefined by its subclasses

```
public abstract class BinaryTreeTraversal {

protected BinaryTree tree;

...

protected Object traverseNode(Position p) {
    TraversalResult r = initResult();
    if (tree.isExternal(p)) {
      external(p, r);
    } else {
      left(p, r);
      r.leftResult = traverseNode(tree.leftChild(p));
      below(p, r);
      r.rightResult = traverseNode(tree.rightChild(p));
      right(p, r);
    }
    return result(r);
  }
```

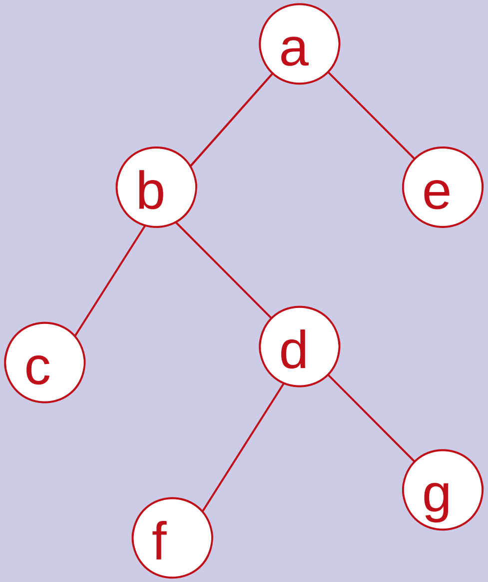
Specializing Generic Binary Tree Traversal

- printing an arithmetic expression

```
public class PrintExpressionTraversal  extends
BinaryTreeTraversal {
...
protected void external(Position p, TraversalResult r)
    { System.out.print(p.element()); }
protected void left(Position p, TraversalResult r)
    { System.out.print("("); }
protected void below(Position p, TraversalResult r)
    { System.out.print(p.element()); }
protected void right(Position p, TraversalResult r)
    { System.out.print(")");  }
}
```

Building tree from pre- and in- order

- Given the preorder and inorder traversals of a binary tree we can uniquely determine the tree.



Preorder	Inorder
a b c d f g e	c b f d g a e
b c d f g	c b f d g
d f g	f d g

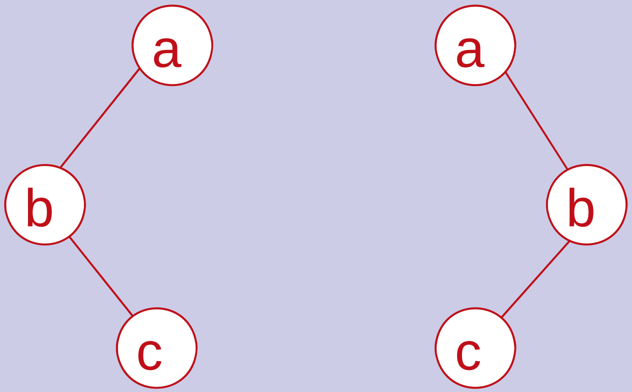
Building tree from post and inorder

- In place of preorder we can use postorder.
- The last node visited in the postorder traversal is the root of the binary tree.
- This can then be used to split in the inorder traversal to identify the left and right subtrees.
- Procedure is similar to the one for obtaining tree from preorder and inorder traversals.

Insufficiency of pre & postorder

- Given the pre and postoder traversal of a binary tree we cannot uniquely identify the tree.
- This is because there can be two trees with the same pre and postorder traversals.

Preorder: a b c
Postorder: c b a



A special case

- If each internal node of the binary tree has at least two children then the tree can be determined from the pre and post order traversals.

Preorder postorder

a b c d f g e   c f g d b e a

b c d f g   c f g d b

d f g   f g d