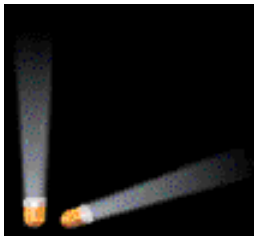
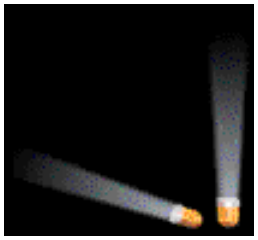
# Binary Tree Traversal Methods





- In a traversal of a binary tree, each element of the binary tree is visited exactly once.
- During the visit of an element, all action (make a clone, display, evaluate the operator, etc.) with respect to this element is taken.

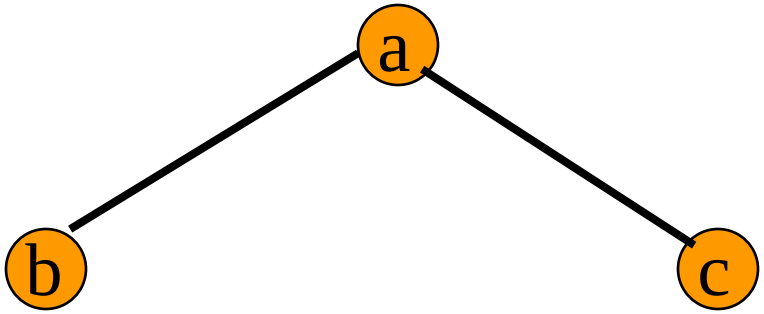
# Binary Tree Traversal Methods

- Preorder
- Inorder
- Postorder
- Level order

# Preorder Traversal

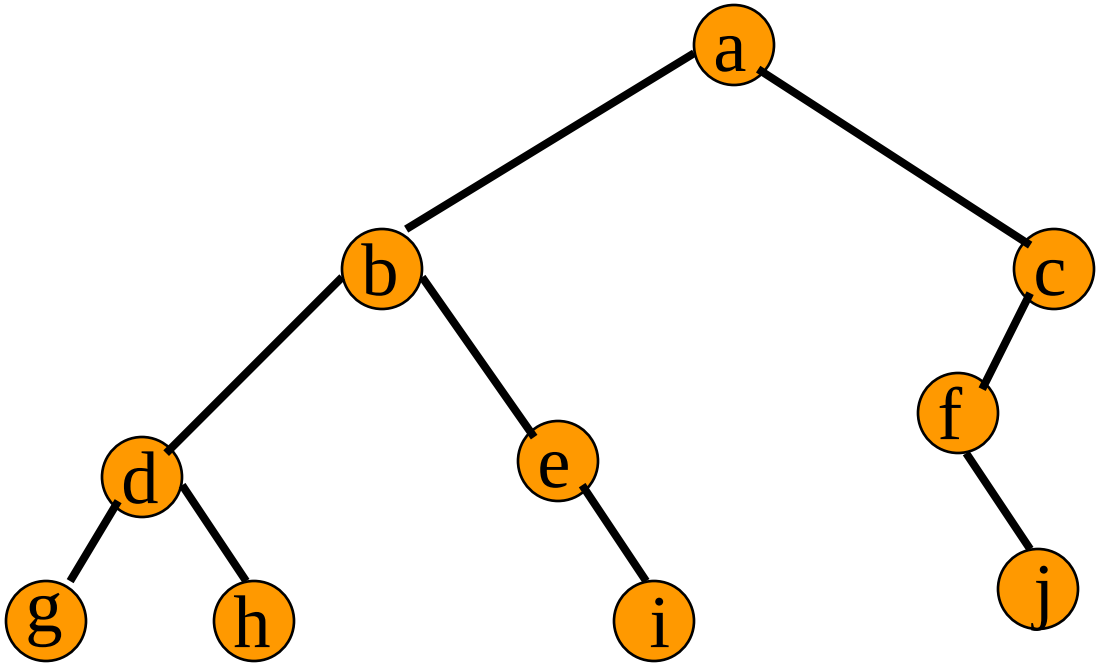
```
void preOrder(treePointer ptr)
{
    if (ptr != NULL)
    {
        visit(t);
        preOrder(ptr->leftChild);
        preOrder(ptr->rightChild);
    }
}
```

# Preorder Example (Visit = print)



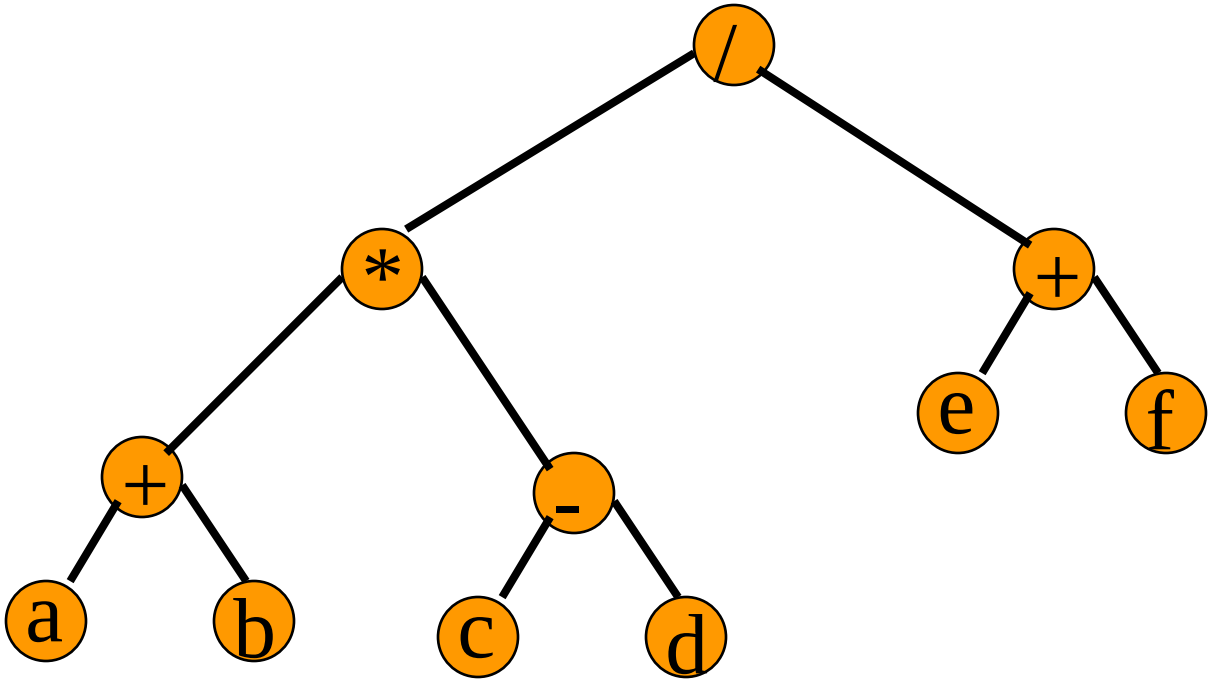

a b c

# Preorder Example (Visit = print)




a b d g h e i c f j

# Preorder Of Expression Tree



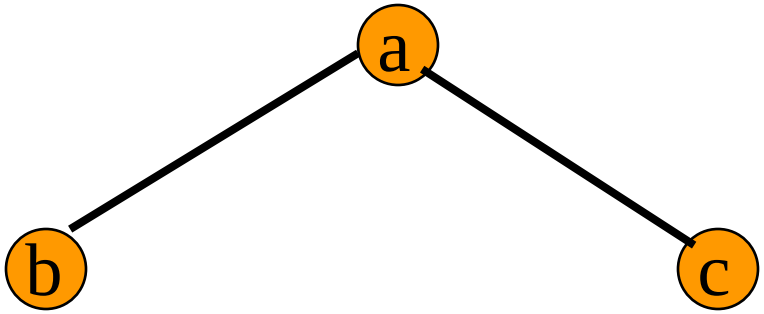

/ * + a b - c d + e f

Gives prefix form of expression!

# Inorder Traversal

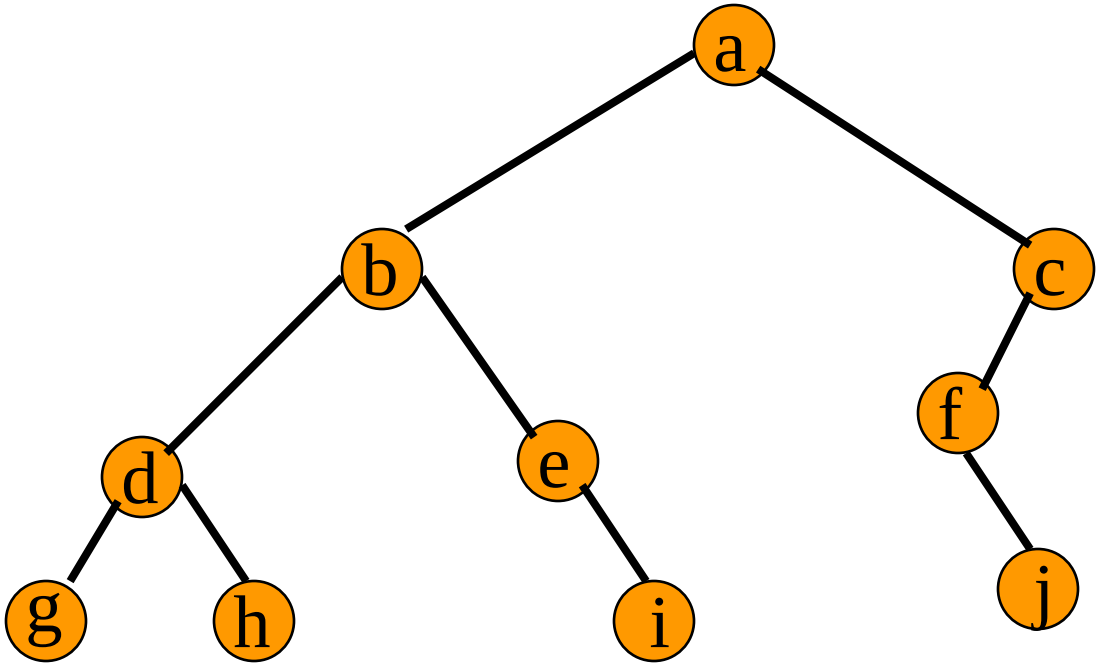
```
void inOrder(treePointer ptr)
{
    if (ptr != NULL)
    {
        inOrder(ptr->leftChild);
        visit(ptr);
        inOrder(ptr->rightChild);
    }
}
```

# Inorder Example (Visit = print)



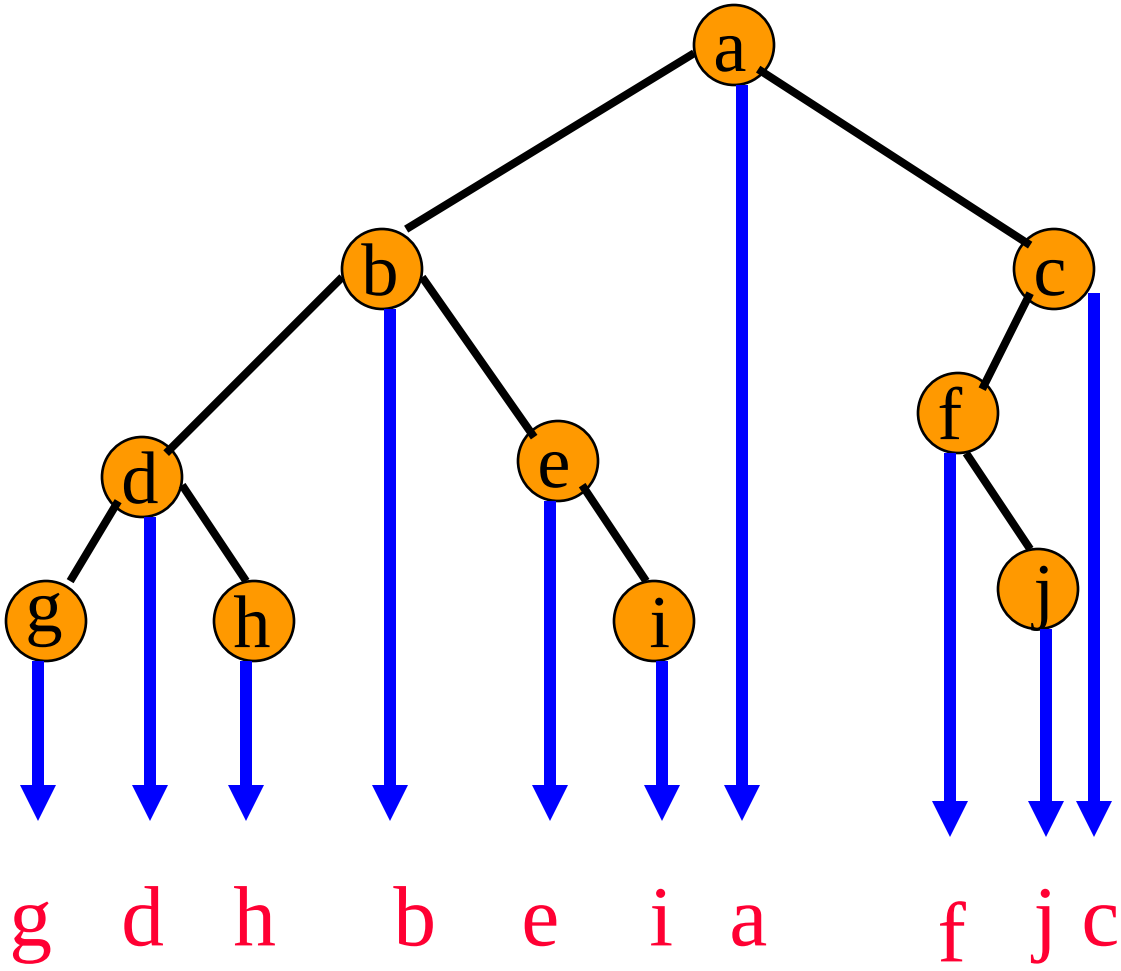

b a c

# Inorder Example (Visit = print)

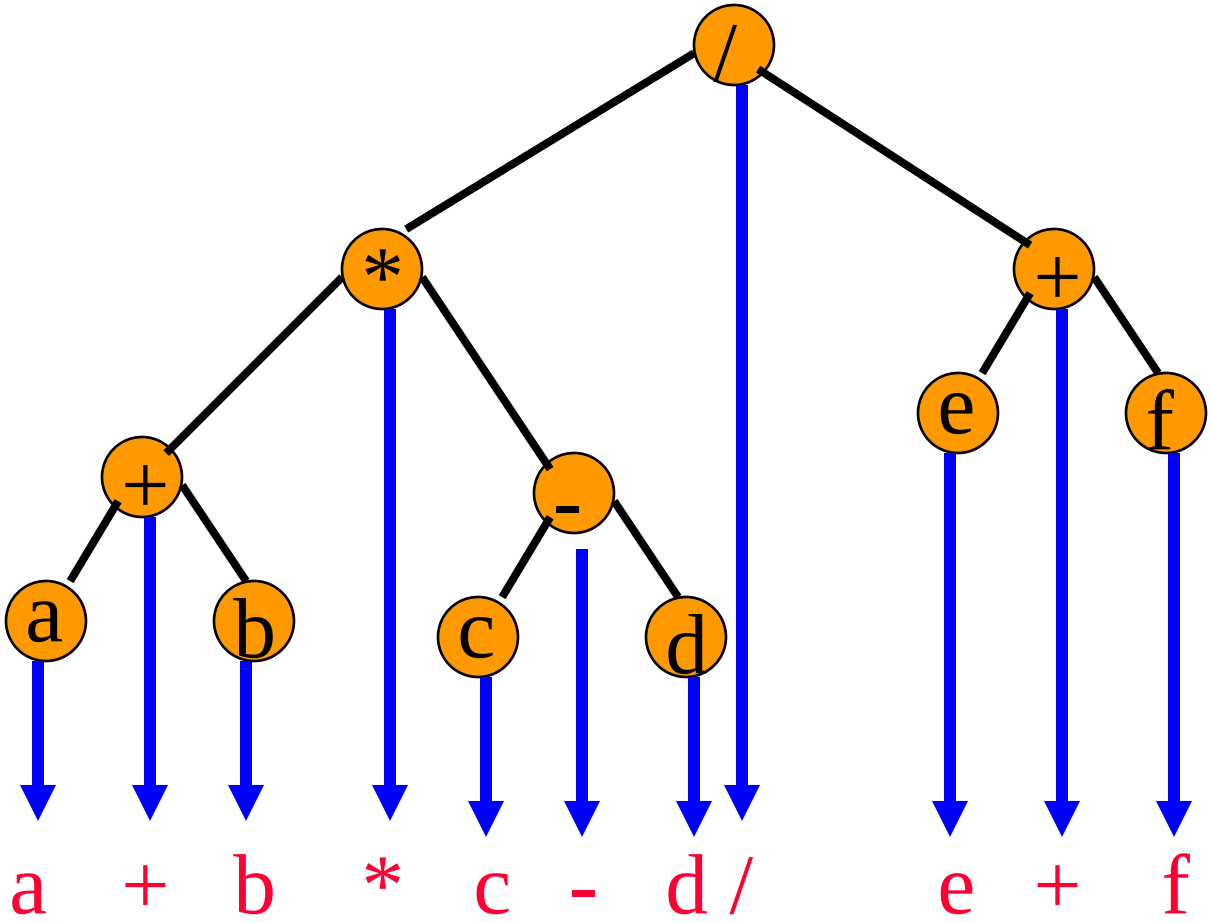



g d h b e i a f j c

# Inorder By Projection (Squishing)



# Inorder Of Expression Tree

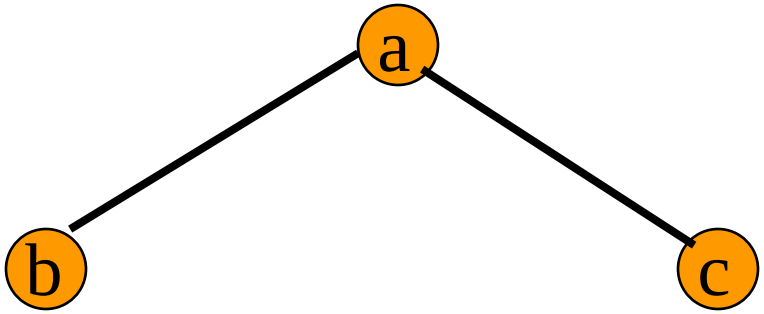



Gives infix form of expression (sans parentheses)!

# Postorder Traversal

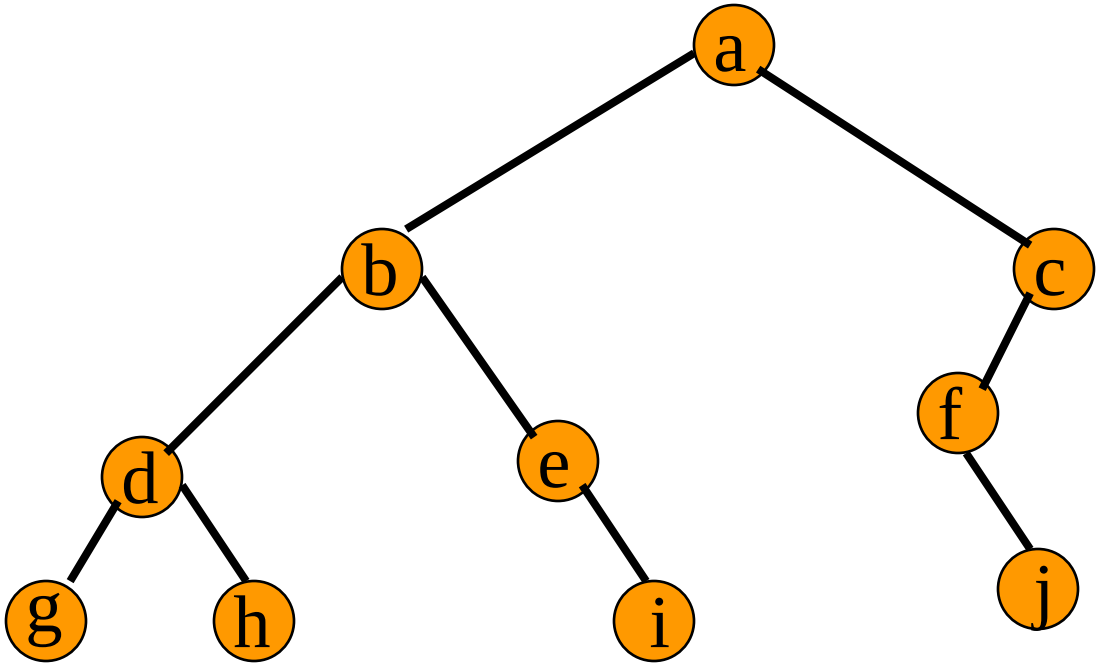
```
void postOrder(treePointer ptr)
{
    if (ptr != NULL)
    {
        postOrder(ptr->leftChild);
        postOrder(ptr->rightChild);
        visit(t);
    }
}
```

# Postorder Example (Visit = print)



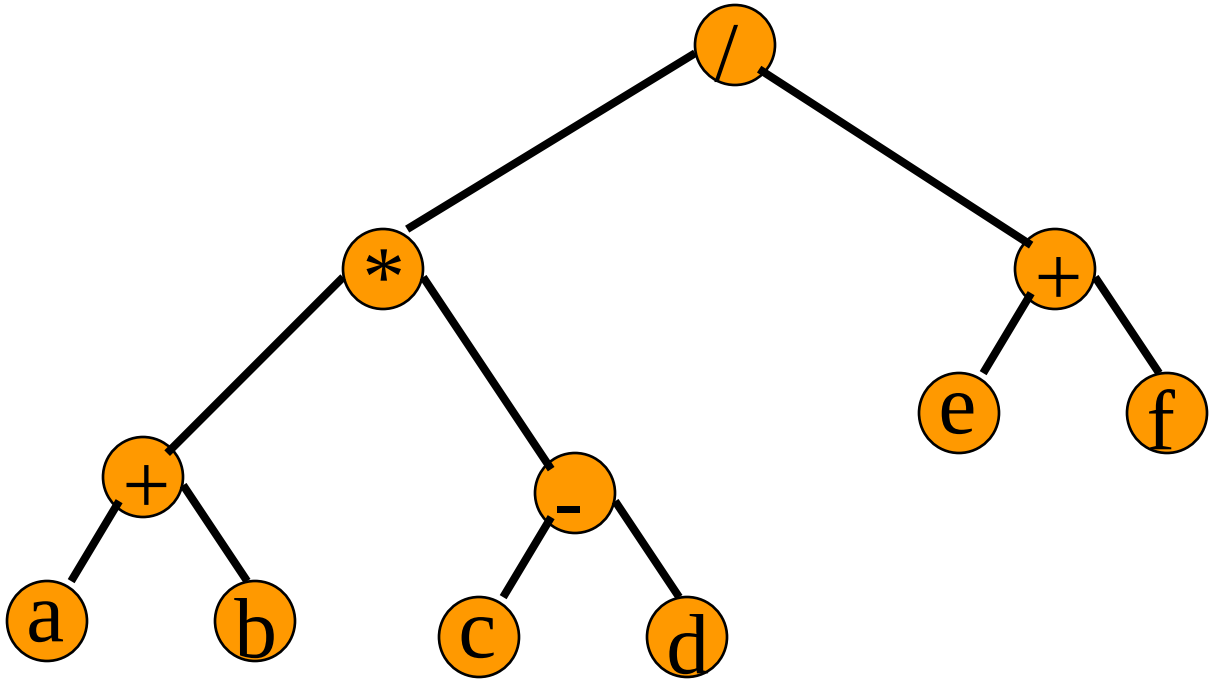

b c a

# Postorder Example (Visit = print)




g h d i e b j f c a

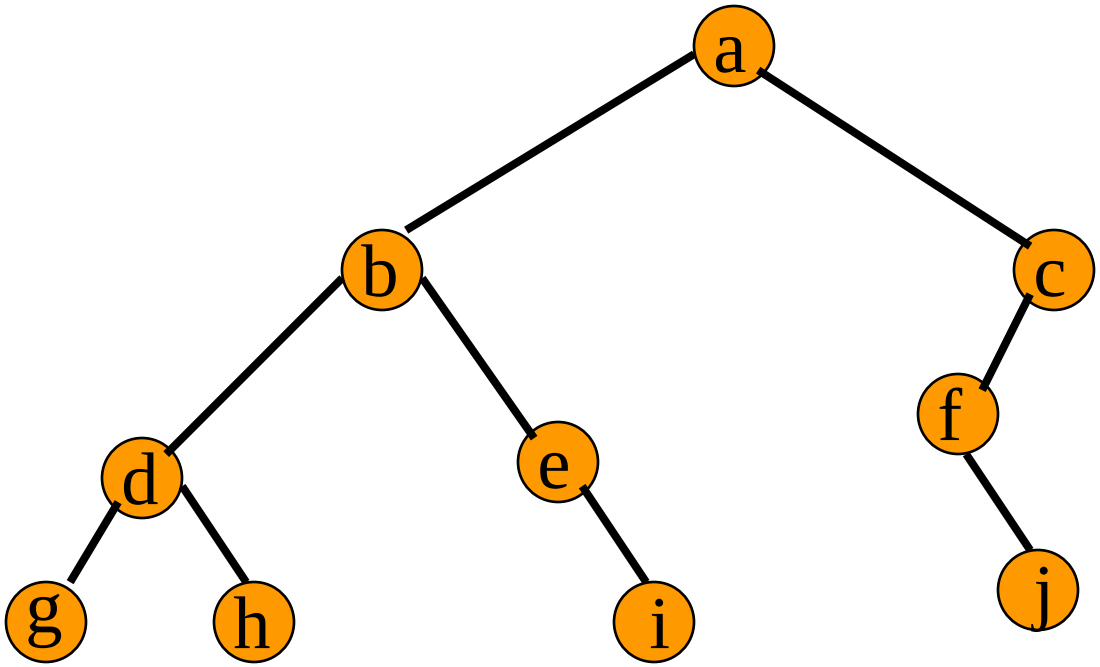
# Postorder Of Expression Tree




a b + c d - * e f + /

Gives postfix form of expression!

# Traversal Applications



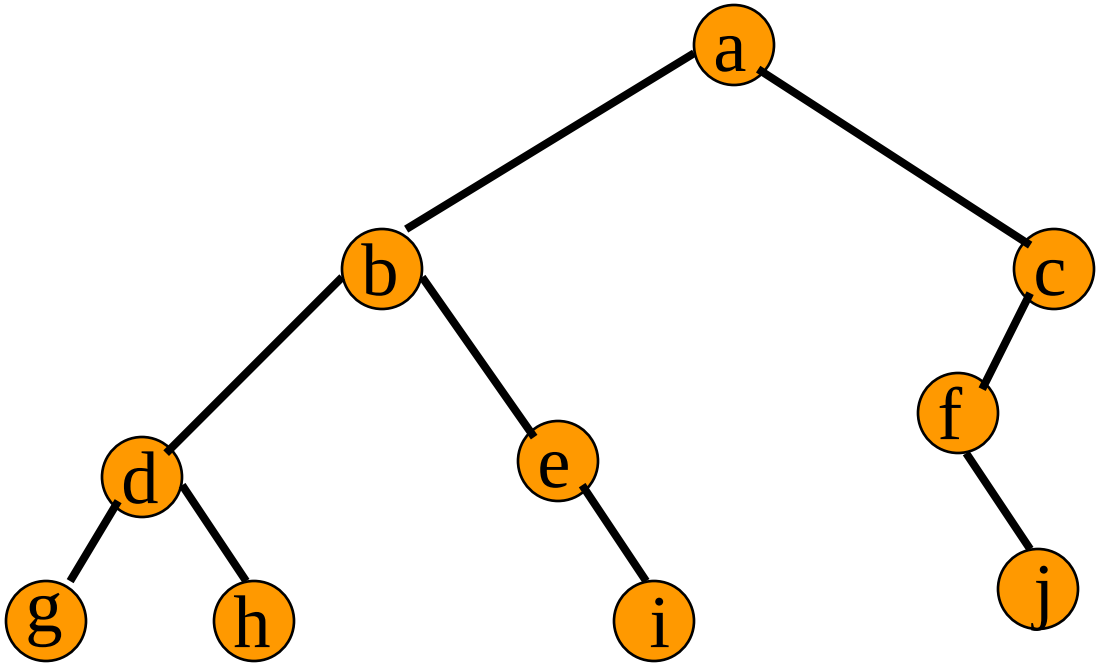

- Make a clone.
- Determine height.
- Determine number of nodes.

# Level Order

Let ptr be a pointer to the tree root.

```
while (ptr != NULL)
{
    visit node pointed at by ptr  and put its children on
    a FIFO queue;
    if FIFO queue is empty, set ptr = NULL;
    otherwise, delete a node from the FIFO queue and
    call it ptr;
}
```

# Level-Order Example (Visit = print)




a b c d e f g h i j

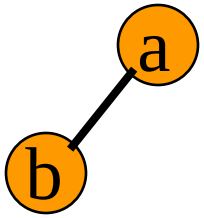
# Binary Tree Construction

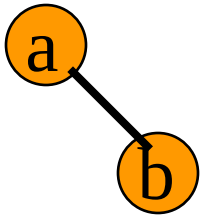
- Suppose that the elements in a binary tree are distinct.
- Can you construct the binary tree from which a given traversal sequence came?
- When a traversal sequence has more than one element, the binary tree is not uniquely defined.
- Therefore, the tree from which the sequence was obtained cannot be reconstructed uniquely.

# Some Examples

preorder

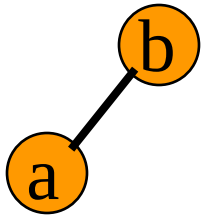
= ab

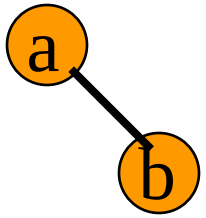



inorder

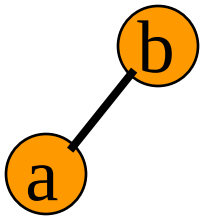
= ab

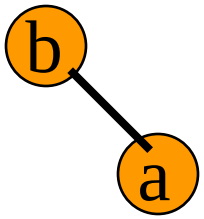



postorder

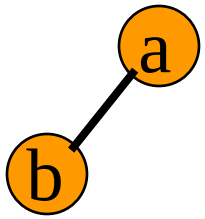
= ab

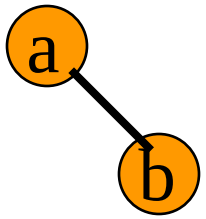



level order

= ab



# Binary Tree Construction

- Can you construct the binary tree, given two traversal sequences?
- Depends on which two sequences are given.

# Preorder And Postorder

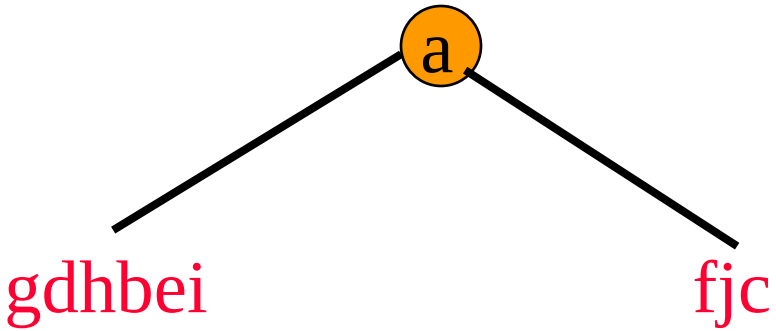
preorder = ab

postorder = ba

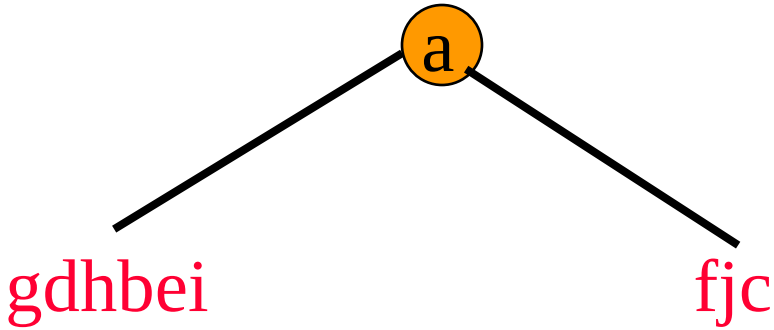

- Preorder and postorder do not uniquely define a binary tree.
- Nor do preorder and level order (same example).
- Nor do postorder and level order (same example).

# Inorder And Preorder

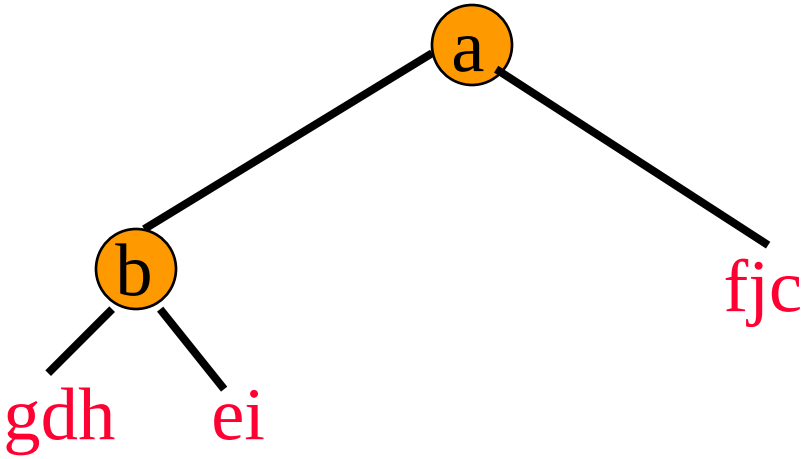
- inorder = g d h b e i a f j c
- preorder = a b d g h e i c f j
- Scan the preorder left to right using the inorder to separate left and right subtrees.
- a is the root of the tree; gdhbei are in the left subtree; fjc are in the right subtree.



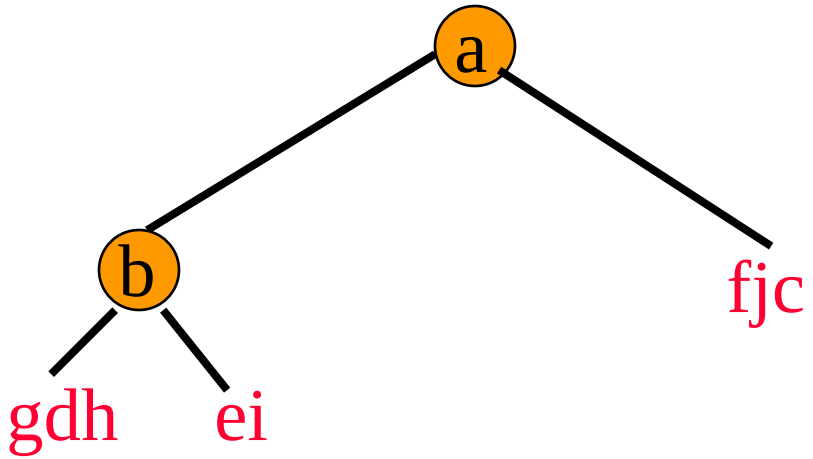
# Inorder And Preorder



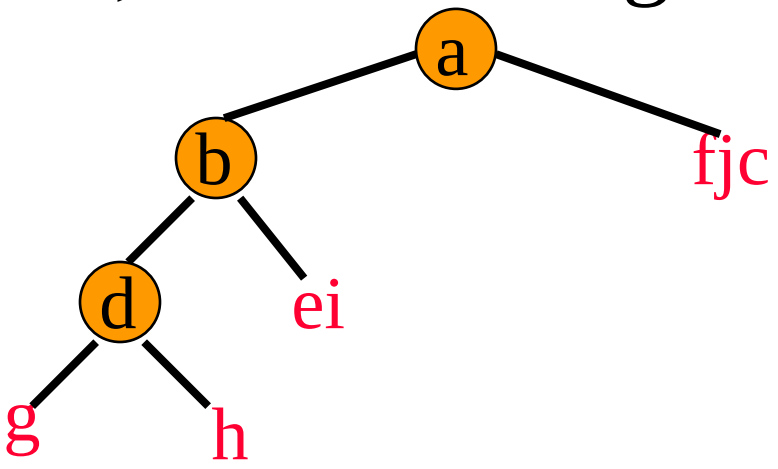

- preorder = a b d g h e i c f j
- b is the next root; gdh are in the left subtree; ei are in the right subtree.



# Inorder And Preorder




- preorder = a b d g h e i c f j
- d is the next root; g is in the left subtree; h is in the right subtree.



# Inorder And Postorder

- Scan postorder from right to left using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- postorder = g h d i e b j f c a
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.

# Inorder And Level Order

- Scan level order from left to right using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- level order = a b c d e f g h i j
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.