Tree Traversals & Expression Tree

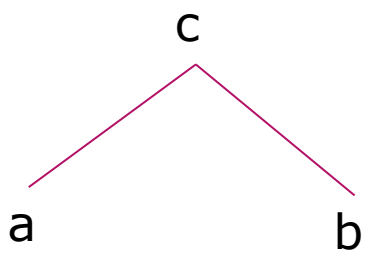
Binary Trees - Traversals

- **Traversals**
  - **Preorder**
  - **Inorder**
  - **Postorder**
- **Expression Tree**

Trees Traversals

- **Traversals**
  - **Visiting the nodes in a tree**
  - **Order of visit ?**

Tree Traversals

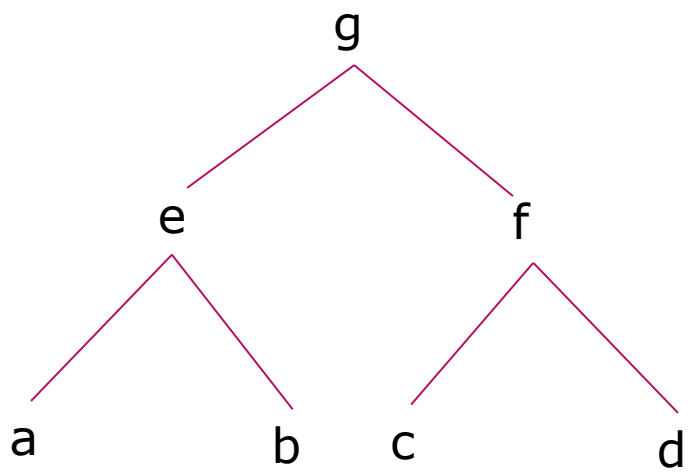


List the nodes in the tree.

a, c, b
a, b, c
c, a, b
c, b, a
……..

In which order to list the nodes?

Tree Traversals



List the nodes.

a, e, b, c, f, d, g
a, b, e, c, d, f, g
……..

In which order ?
Is there any standard order?

Example - Expression Tree

- **Language Translator (Compiler / Interpreter)**
  - **Steps in translating x = a+b**
  - **Semantics of x = a+b ?**
  - **In general, statements of the form *identifier = <expression>***

Translation – Assignment Statement

- **Language Translator - Interpreter**
  - **Steps in translating x = a+b**
- **Semantics of x = a+b ?**
  - **Evaluate a+b, store the value in variable x**
- **In general for *variable = <expression>***
  - **Evaluate r.h.s *expression,* store value in the l.h.s *variable***

Expression Evaluation

- **Expressions**
  - **a+b*c**
  - **(a+b*c+d) + (e/f+b*c+d%g-h)**
  - **x && y || p && q**
- **Order of evaluation of subexpressions**
  - **Based on operator precedence and associativity**

Expression Evaluation- precedence

- **a+b*c**
  - * has higher precedence than +
  - b*c to be evaluated first
- **Parenthesizing**
  - **a+(b*c)**

Expression Evaluation-associativity

- **a+b+c**
  - **+ is left associative**
  - **a+b to be evaluated first**
- **Parenthesizing**

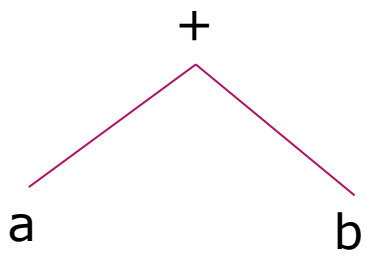
  (a+b) + c

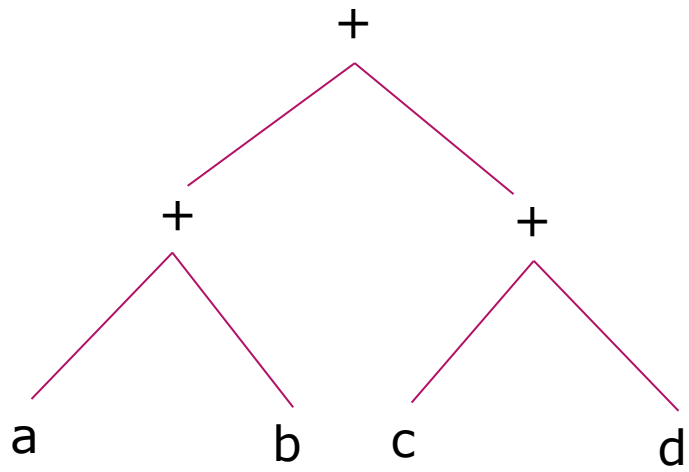
Expression Evaluation – order of evaluation of subexpressions

- **Order of evaluation of subexpressions**
  - **Based on operator precedence and associativity**

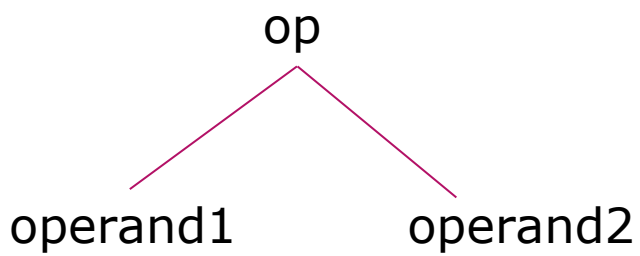
    - **(a+b)*(c+d)**
    - **a+b*c+d**
    - **(a+b*c+d) + (e/f+b*c+d%g-h)**
    - **x && y || p && q**

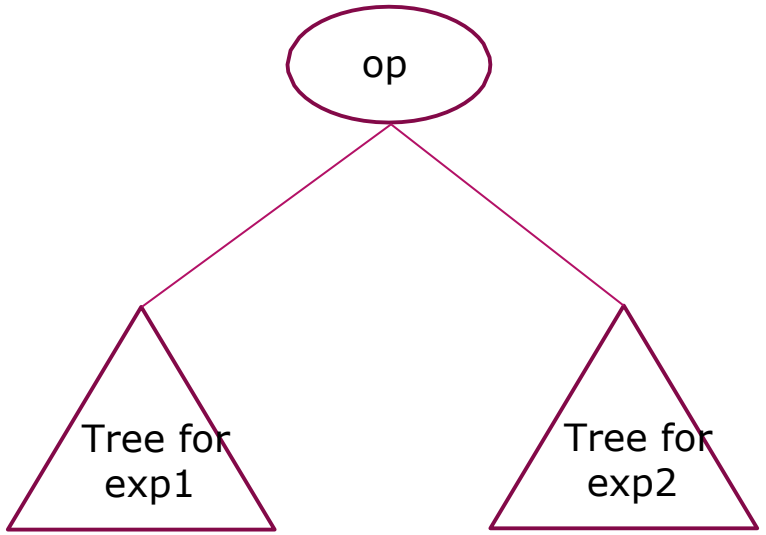
Expression Tree





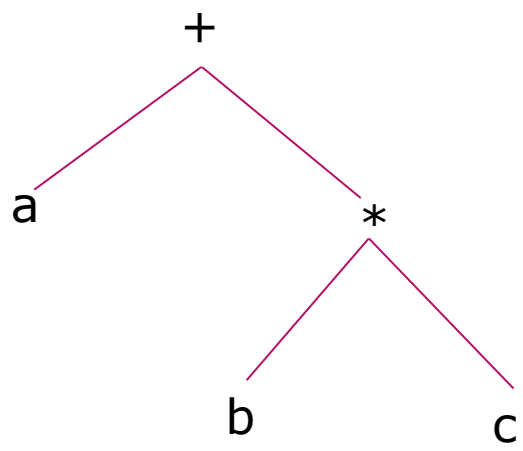
Expression Tree



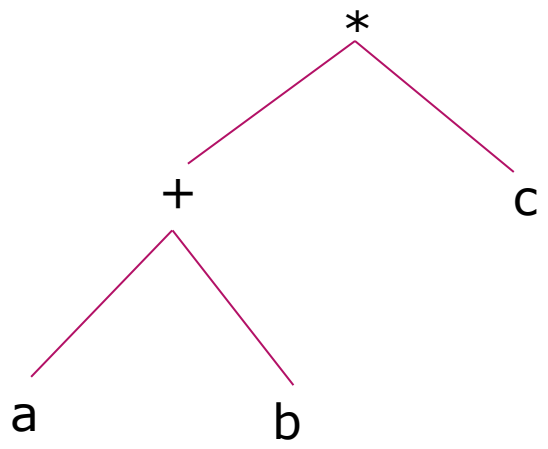


Tree for exp1 op exp2

Expression Tree



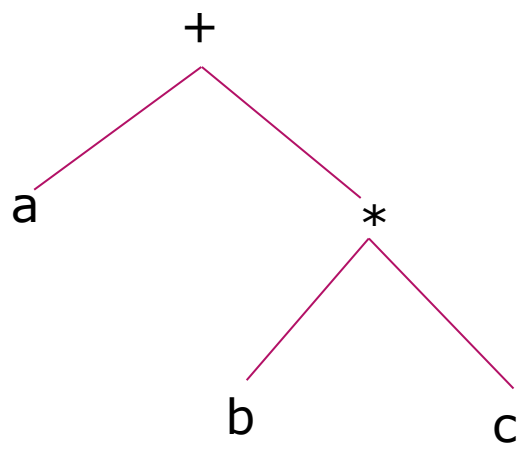
a+(b*c)



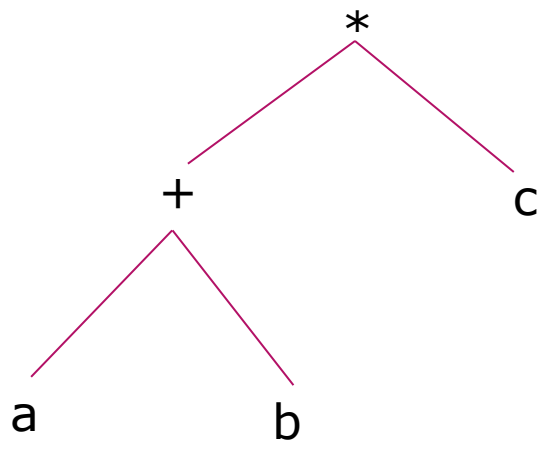
(a+b)*c

Order of evaluation ?

Expression Tree



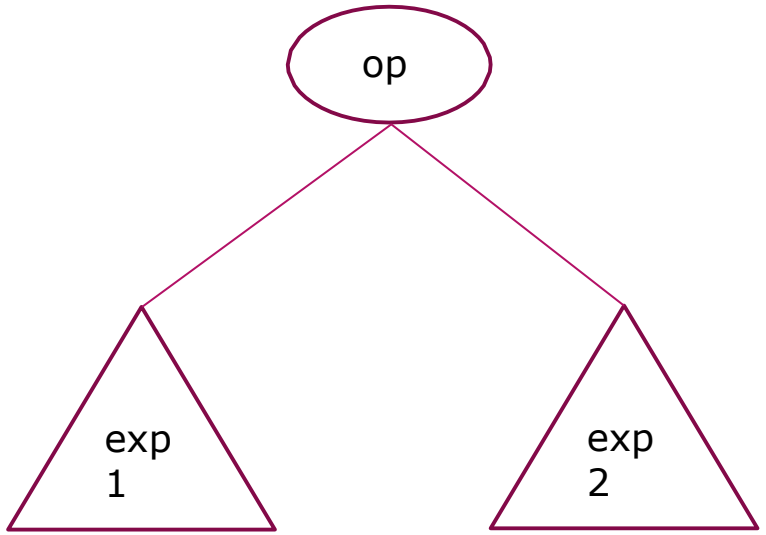
a+(b*c)



(a+b)*c

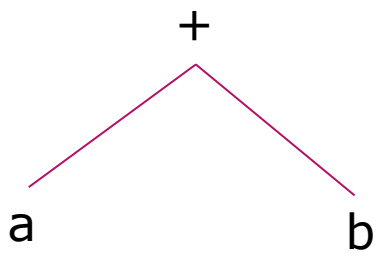
Order of evaluation : Evaluate the subtrees first

Expression Tree - evaluation



Evaluate exp1 □ v1
Evaluate exp2 □ v2
Evaluate v1 op v2

Expression Tree – listing nodes

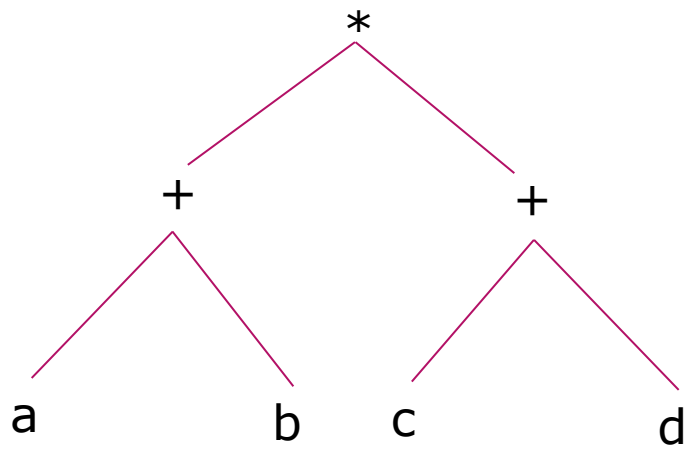


+ab root, lchild, rchild

a+b lchild, root, rchild

ab+ lchild, rchild, root

Expression Tree – listing nodes

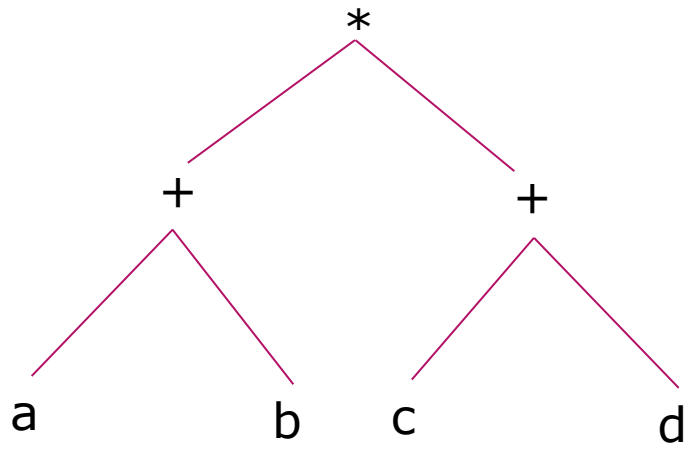


***+ab+cd** root, left subtree, right subtree

a+b*c+d left subtree, root, right subtree

ab+cd+* left subtree, right subtree, root

Tree Traversals



***+ab+cd**	**preorder**	**D L R**
a+b*c+d	**inorder**	**L D R**
ab+cd+*	**postorder**	**L R D**

Tree Traversals

- **Preorder**
  - **Root, Left subtree in preorder, Right subtree in preorder**
- **Inorder**
  - **Left subtree in inorder, Root, Right subtree in inorder**
- **Postorder**
  - **Left subtree in postorder, Right subtree in postorder, root**

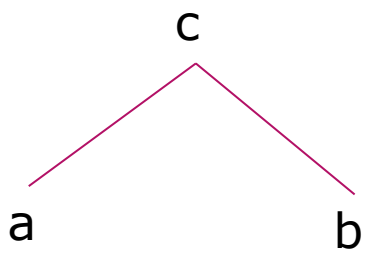
Tree Traversals

- **Algorithms**
  - **Recursive – direct and easy**
  - **Iterative – using Stack**

Tree Traversals

- **Tree Traversal (Tree Walk in CLRS)**
- **Preorder**
  - **Visit Root, Left subtree in preorder, Right subtree in preorder**
- **Inorder**
  - **Left subtree in inorder, Visit Root, Right subtree in inorder**
- **Postorder**
  - **Left subtree in postorder, Right subtree in postorder, visit Root**

Tree Traversals

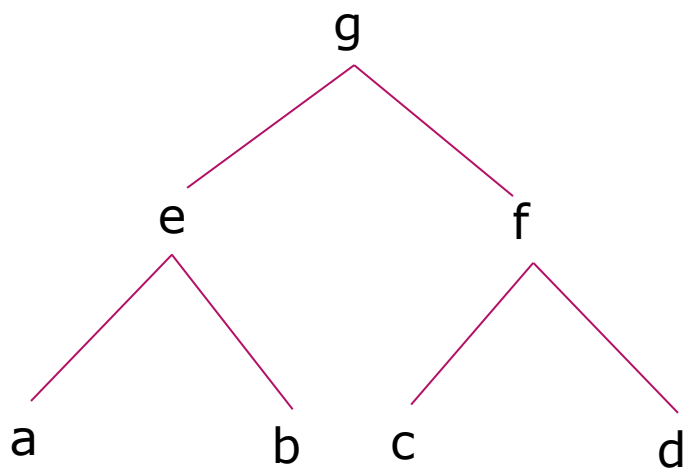


Preorder: c a b

Inorder: a c b

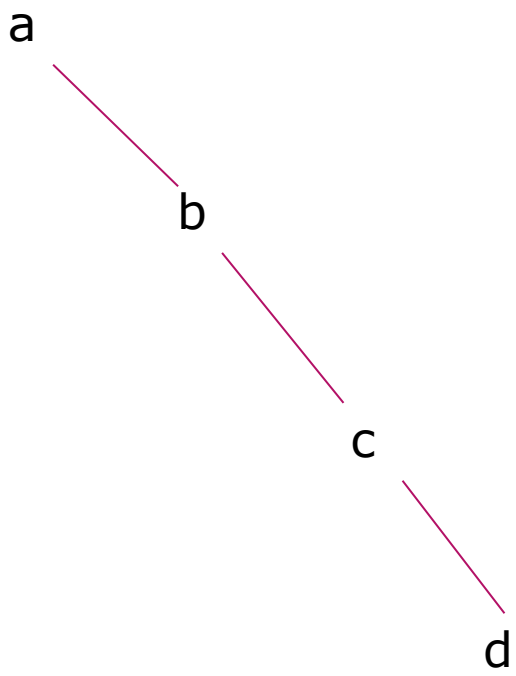
Postorder: a b c

Tree Traversals



List the nodes in preorder, inorder, postorder

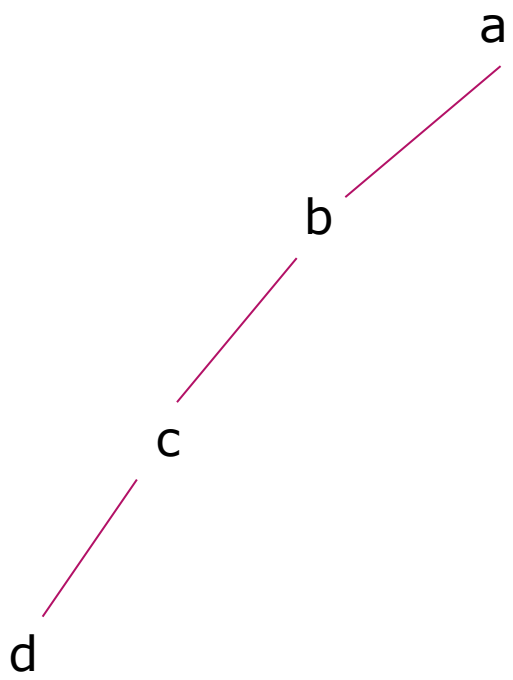
Tree Traversals



List the nodes in preorder, inorder, postorder

Skewed Tree (Right skewed)

Tree Traversals



List the nodes in preorder, inorder, postorder

Inorder Tree Walk - Algorithm

```
INORDER-TREE-WALK (x)
    if x ≠NIL
        INORDER-TREE-WALK (x.left)
        print x.data
        INORDER-TREE-WALK (x.right)
```

Inorder Tree Walk - Algorithm

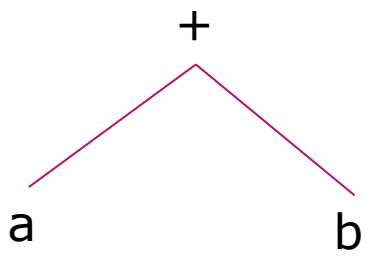
```
INORDER-TREE-WALK (x)
    if x ≠NIL
        INORDER-TREE-WALK (x.left)
        print x.data
        INORDER-TREE-WALK (x.right)
```

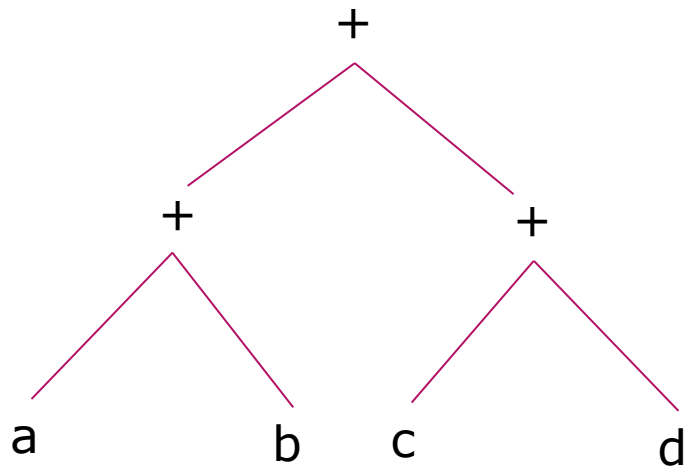
- x is a node in the tree
- To traverse the entire tree T, invoke as INORDER-TREE-WALK (T.root)

Tree Walk - Algorithms

- **Write recursive algorithms for**
  - `PREORDER-TREE-WALK()`
  - `POSTORDER-TREE-WALK()`

Exercise: Expression Tree - Traversals





Exercise: Expression Tree - Traversal