Ch 11.3 Tree Traversal Algorithms

- Traversal algorithm

  - procedure for systematically visiting every vertex of an ordered rooted tree

- Tree traversals are defined recursively
- Three traversals

  preorder

  inorder

  postorder

UTD

Tree Traversal Algorithms

PREORDER Traversal Algorithm

- Let 'T' be an ordered binary tree with root r
- If T has only r, then r is the preorder traversal
- Otherwise, suppose T_1, T_2 are the left and right sub-trees at r
- The preorder traversal begins by visiting r
- Then traverses T_1 in preorder, then traverses T_2 in preorder

Preorder Traversal

Step 1: Visit r

Step 2: Visit T_1 in preorder

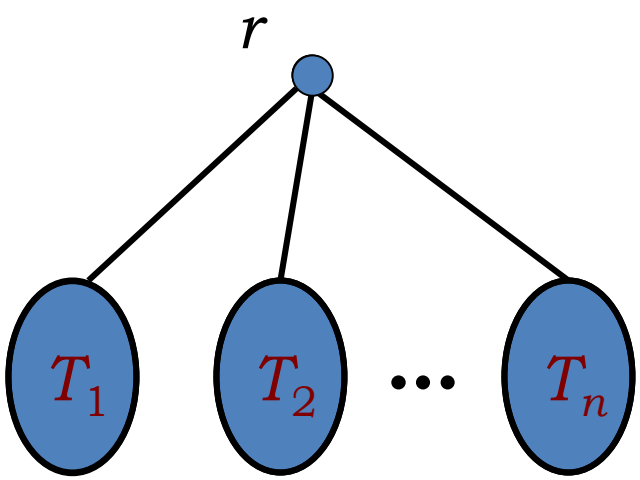
Step 3: Visit T_2 in preorder

.

.

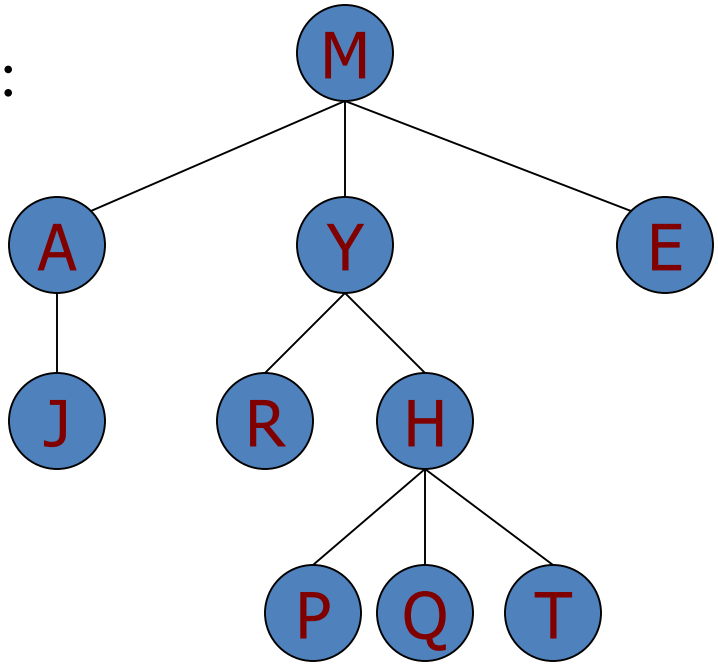
.

Step n+1: Visit T_n in preorder



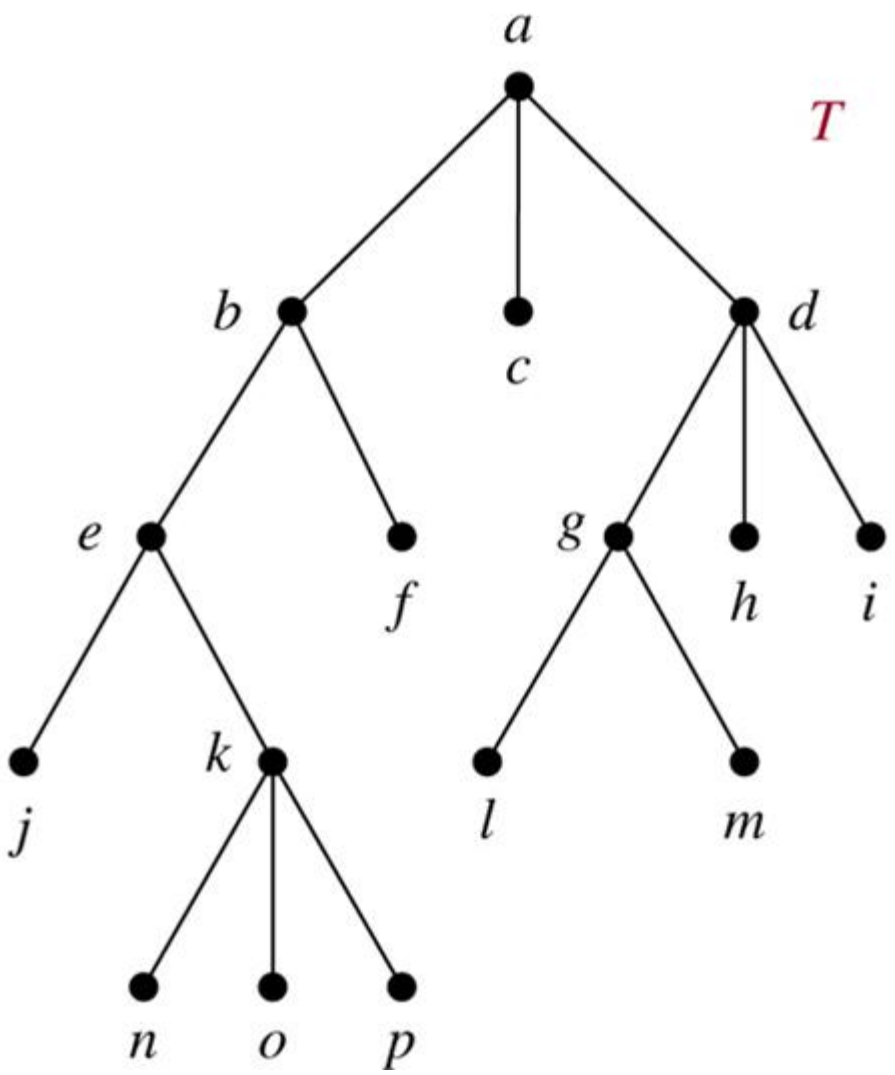
Example

Tree:



Visiting sequence:

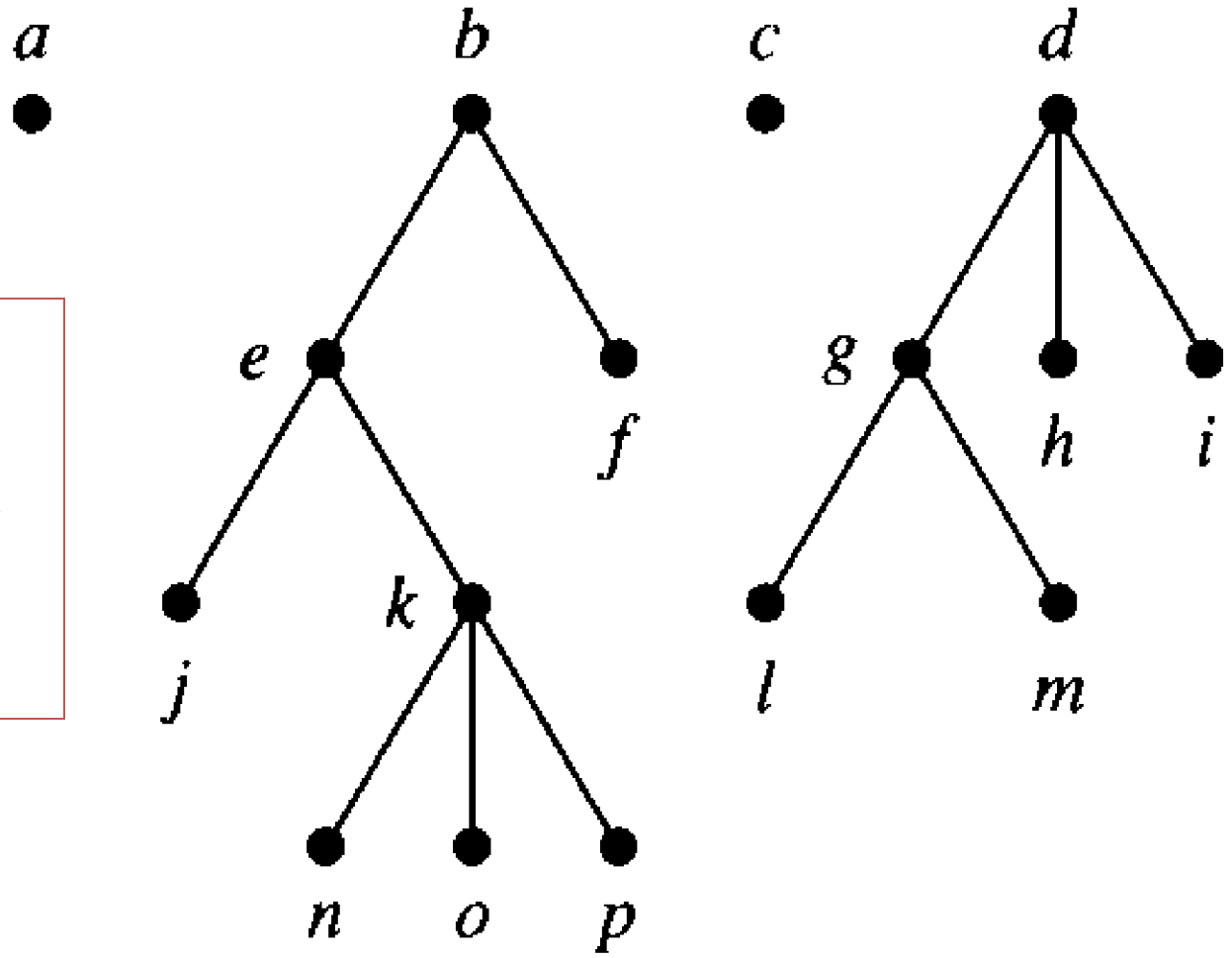
The Preorder Traversal of *T*



In which order does a preorder traversal visit the vertices in the ordered rooted tree *T* shown to the left?

Preorder:
Visit root, then visit subtrees left to right.

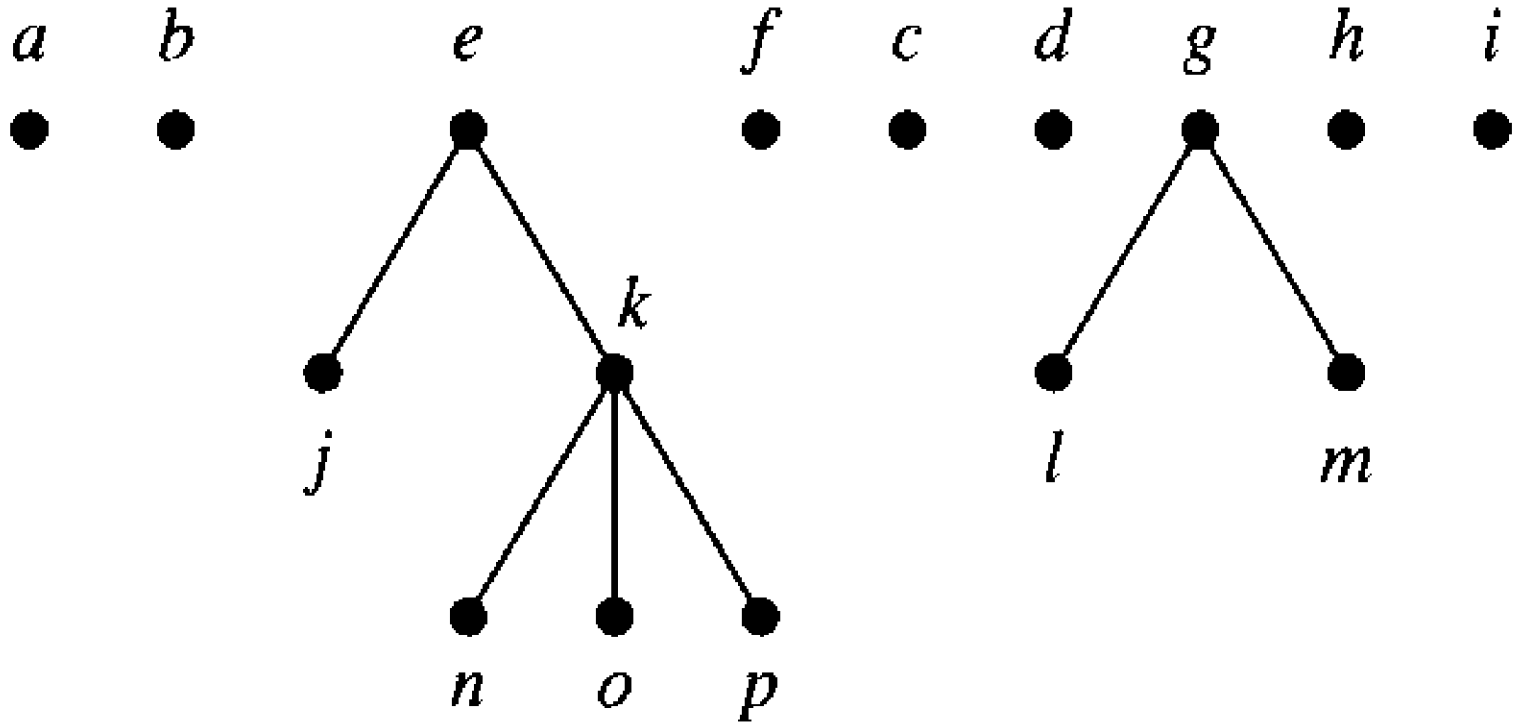
The Preorder Traversal of *T*



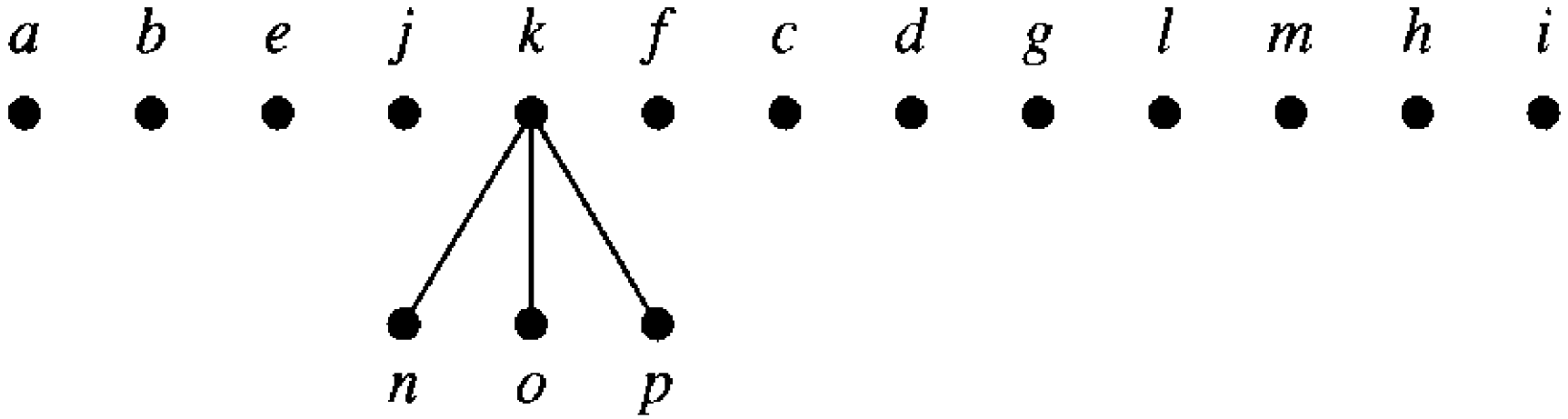
Preorder:

Visit root, then visit subtrees left to right.

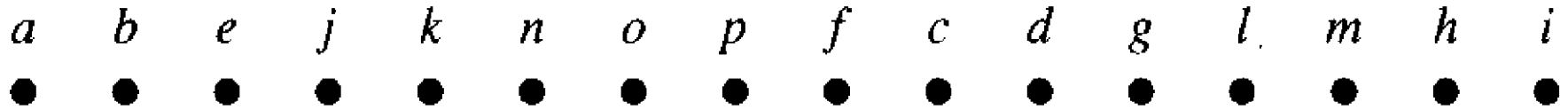
The Preorder Traversal of *T*



The Preorder Traversal of *T*



The Preorder Traversal of *T*



Tree Traversal Algorithms

INORDER Traversal Algorithm

- Let T be an ordered rooted tree with root r
- If T has only r, then r is the inorder traversal
- Otherwise, suppose T_1, T_2 are the left and right sub-trees at r
- The in-order traversal begins by traversing T_1 in inorder
- Then visits r, then traverses T_2 in in-order

Inorder Traversal

Step 1: Visit T_1 in inorder

Step 2: Visit r

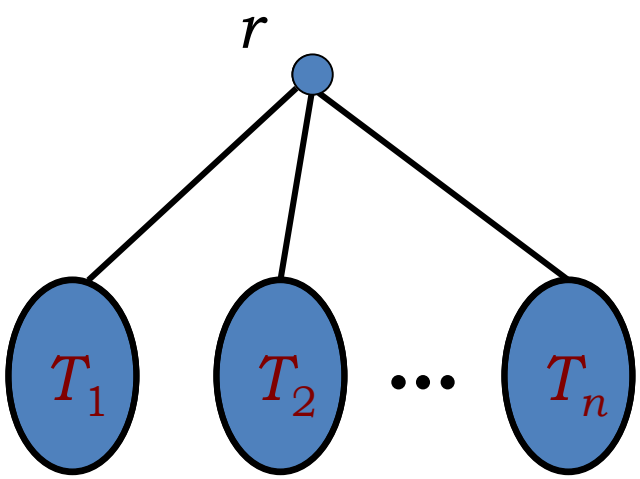
Step 3: Visit T_2 in inorder

.

.

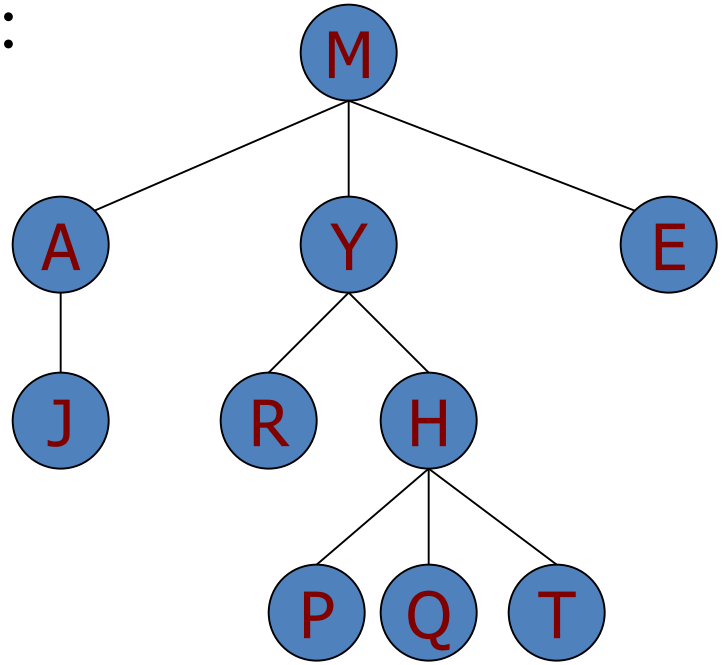
.

Step n+1: Visit T_n in inorder



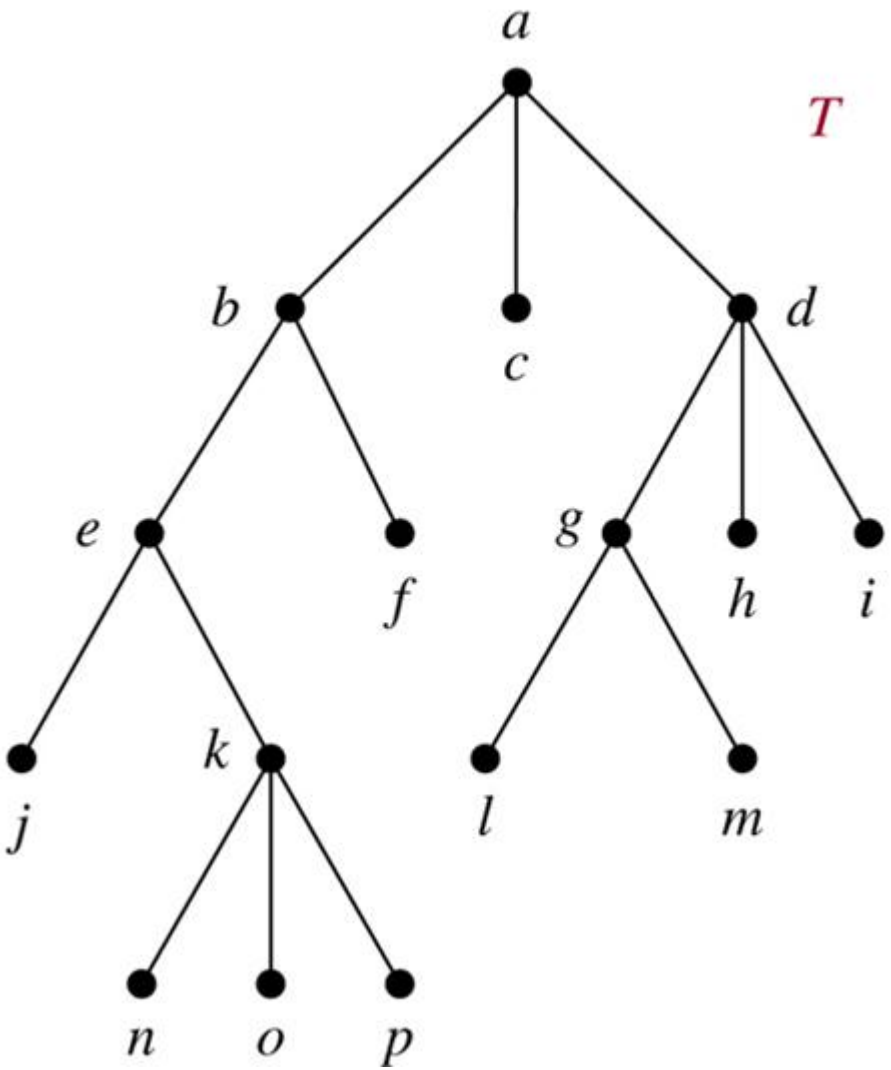
Example

Tree:



Visiting sequence:

The Inorder Traversal of *T*

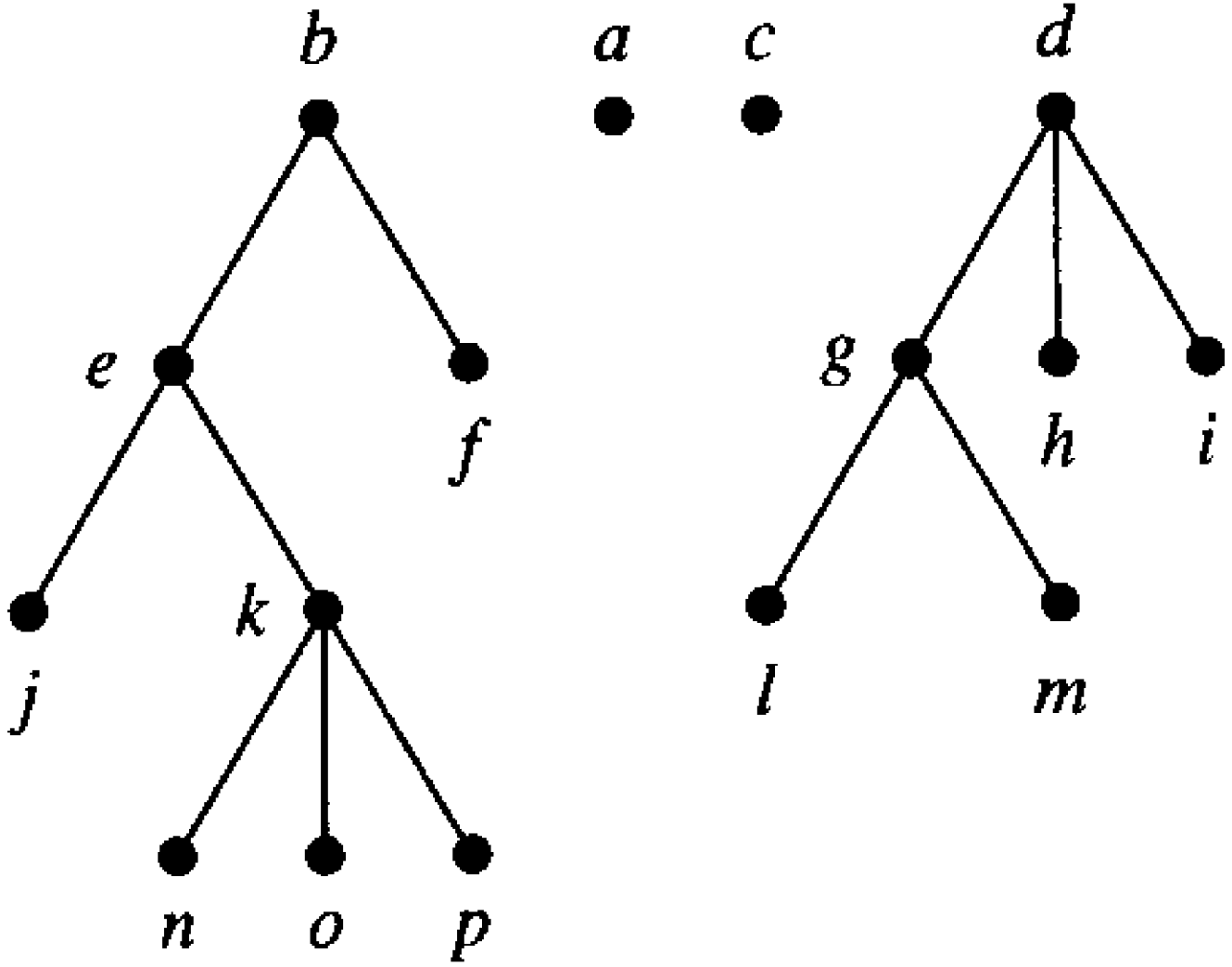


In which order does an inorder traversal visit the vertices in the ordered rooted tree *T* shown to the left?

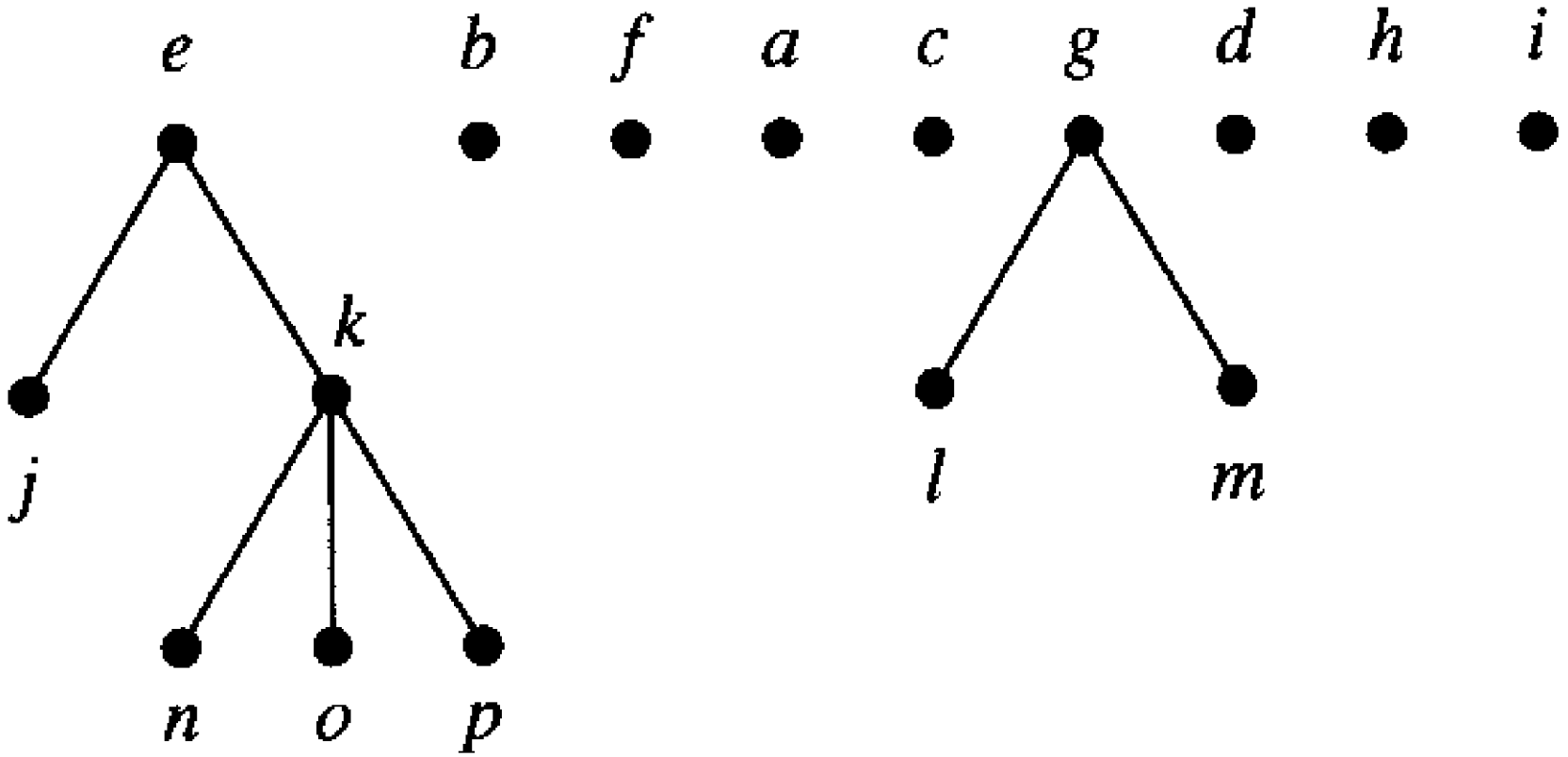
Inorder:
Visit leftmost tree, visit root, visit other subtrees left to right.

The Inorder Traversal of *T*

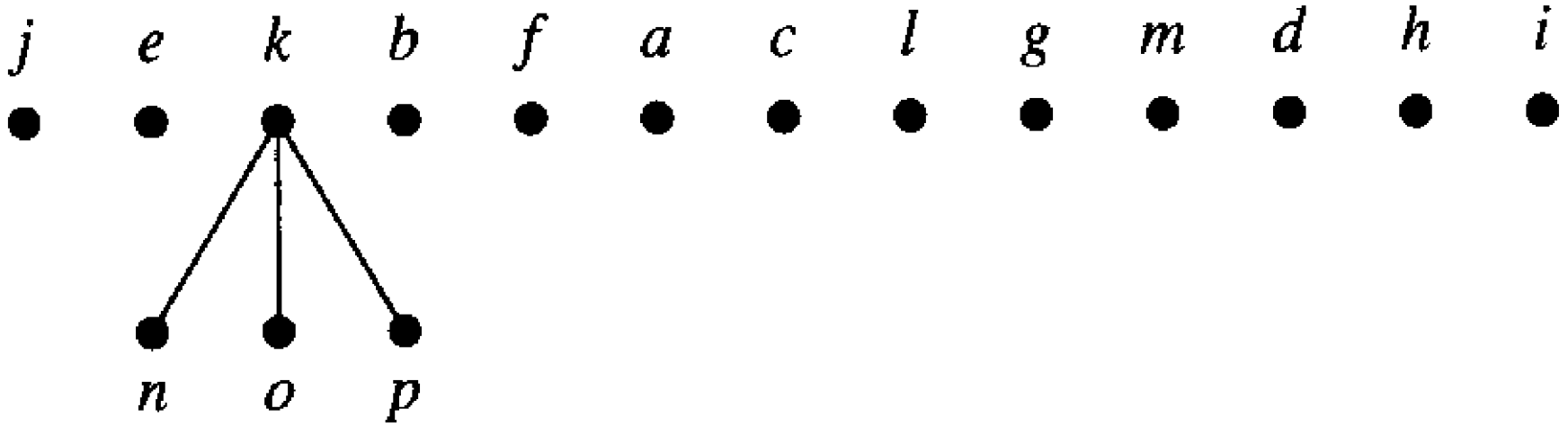
Inorder:
Visit
leftmost tree,
visit root,
visit other
subtrees left
to right.



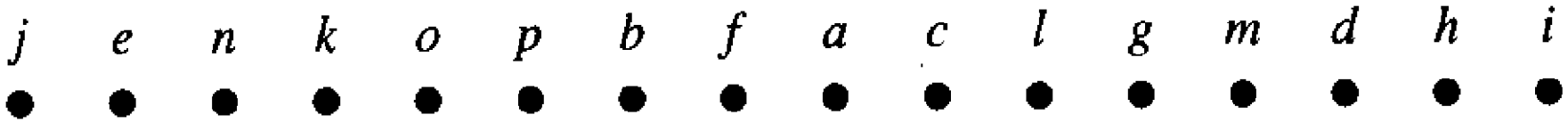
The Inorder Traversal of *T*



The Inorder Traversal of *T*



The Inorder Traversal of *T*



Tree Traversal Algorithms

POSTORDER Traversal Algorithm

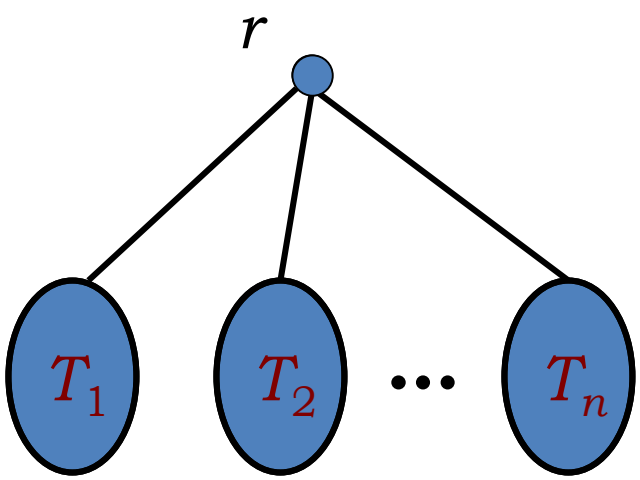
- Let T be an ordered rooted tree with root r
- If T has only r, then r is the postorder traversal
- Otherwise, suppose T_1, T_2 are the left and right subtrees at r
- The postorder traversal begins by traversing T_1 in postorder
- Then traverses T_2 in postorder, then ends by visiting r

Postorder Traversal

Step 1: Visit T_1 in postorder
Step 2: Visit T_2 in postorder

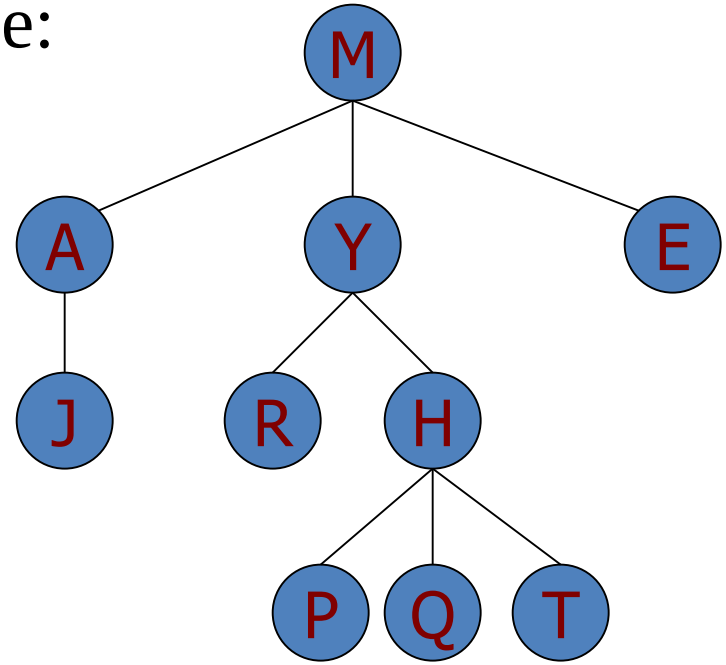
.
.
.

Step n: Visit T_n in postorder
Step n+1: Visit r



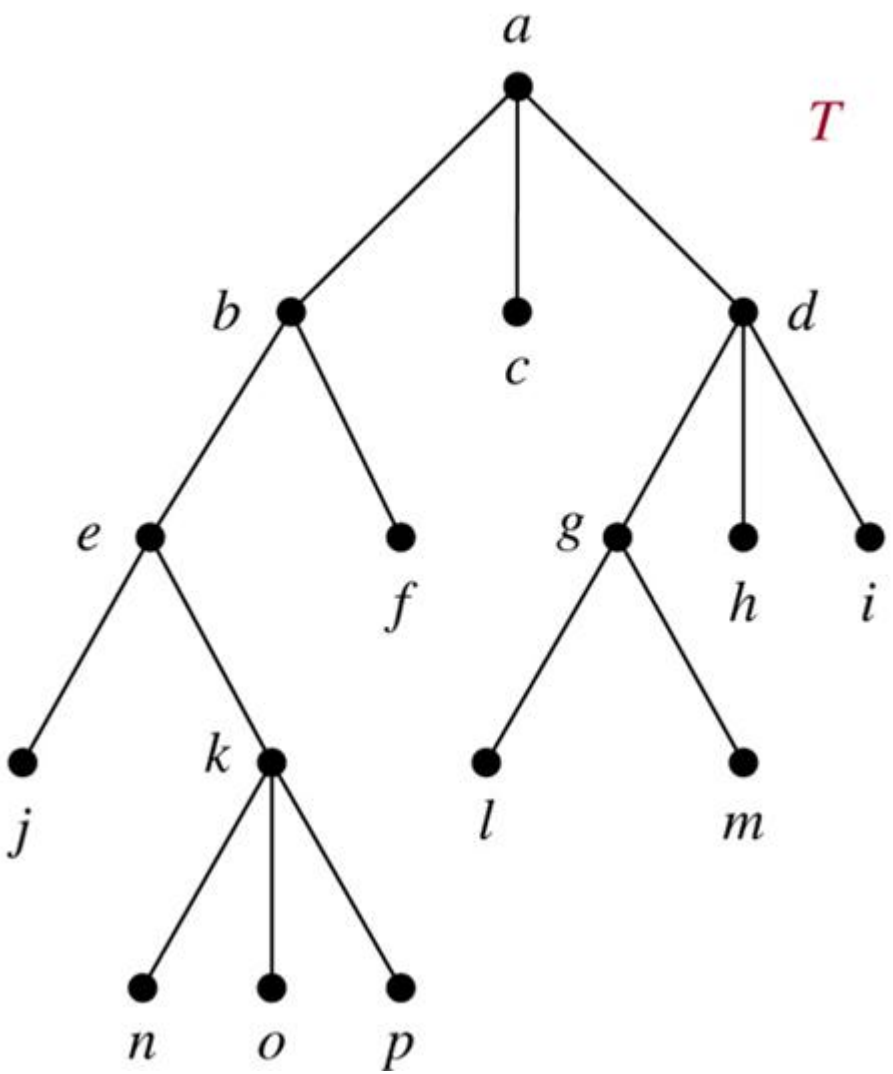
Example

Tree:



Visiting sequence:

The Postorder Traversal of *T*

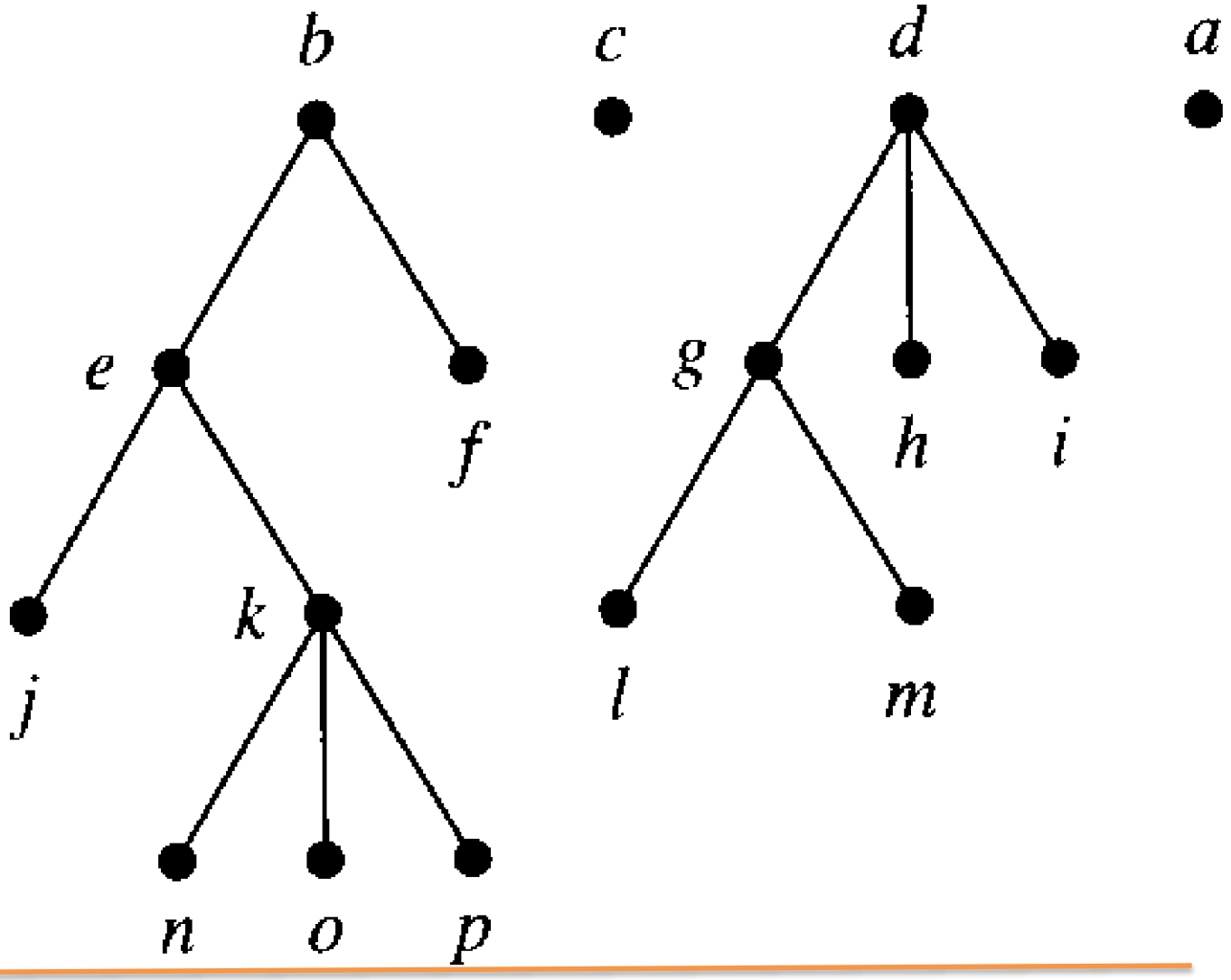


In which order does a postorder traversal visit the vertices in the ordered rooted tree *T* shown to the left?

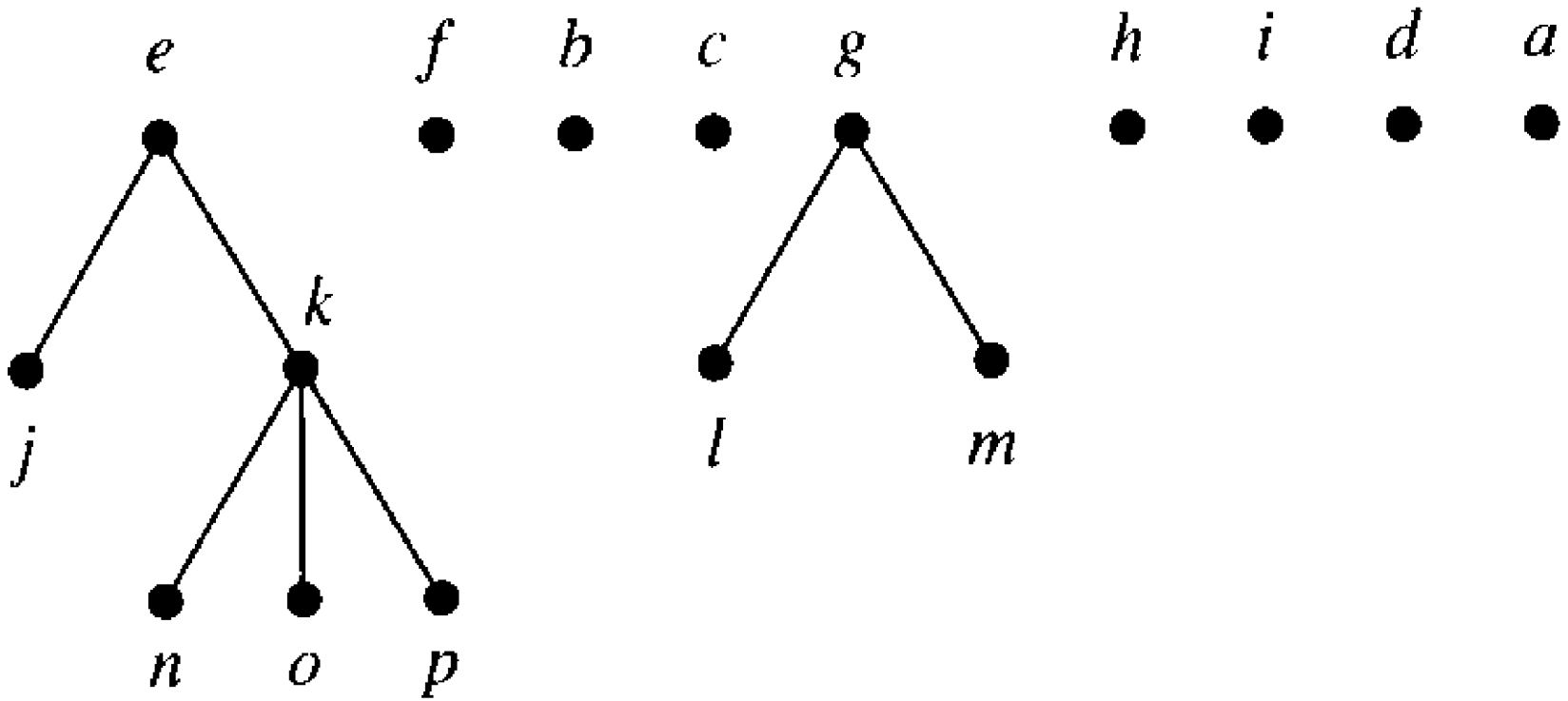
Postorder:
Visit subtrees left to right, then visit root.

The Postorder Traversal of *T*

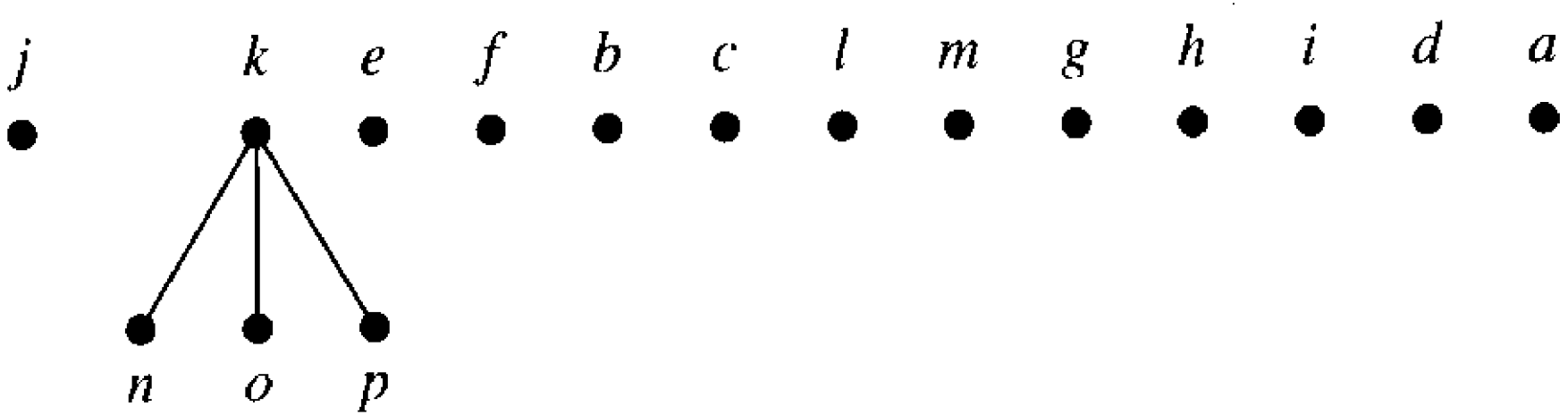
Postorder: Visit subtrees left to right, then visit root.



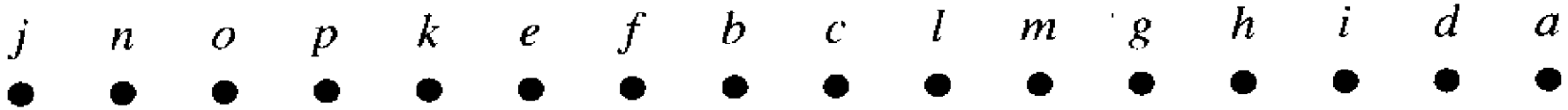
The Postorder Traversal of *T*



The Postorder Traversal of *T*



The Postorder Traversal of *T*



Representing Arithmetic Expressions

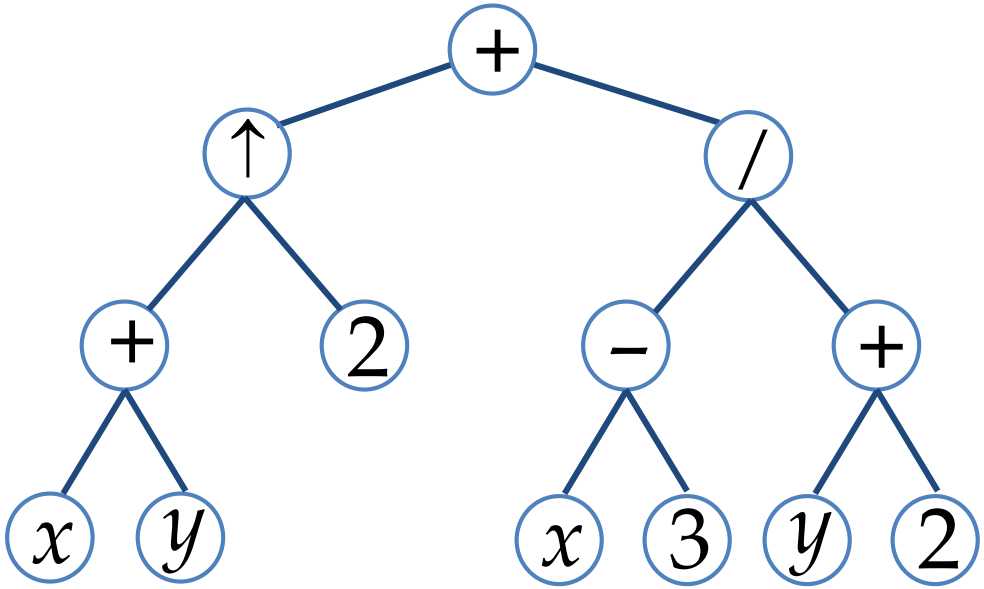
- Complicated arithmetic expressions can be represented by an ordered rooted tree
  - Internal vertices represent operators
  - Leaves represent operands
- Build the tree bottom-up
  - Construct smaller subtrees
  - Incorporate the smaller subtrees as part of larger subtrees

Example

$$(x+y)^2 + (x-3)/(y+2)$$

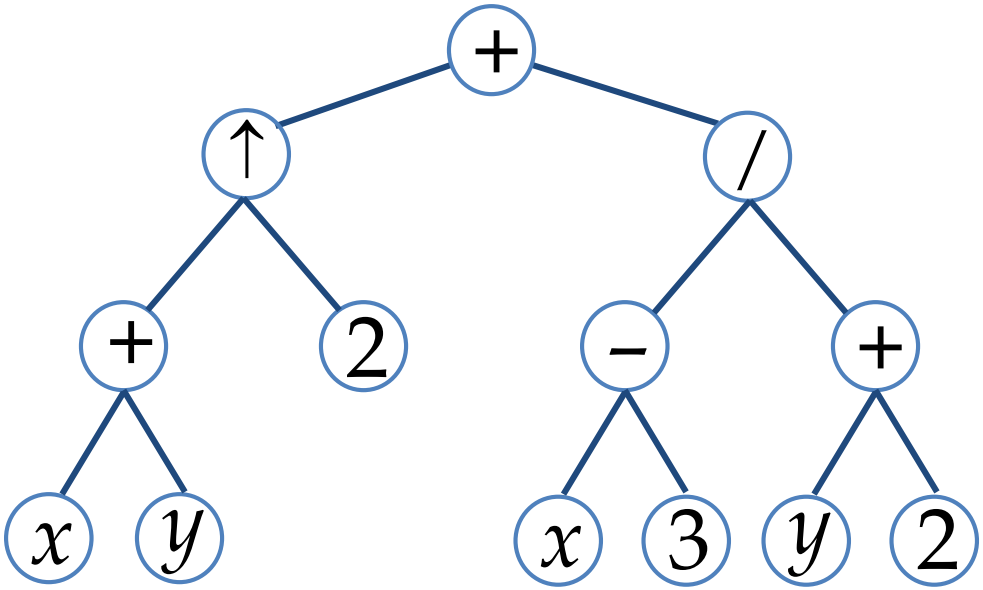
Infix Notation

- Traverse in inorder adding parentheses for each operation



Prefix Notation (Polish Notation)

- Traverse in preorder:



Evaluating Prefix Notation

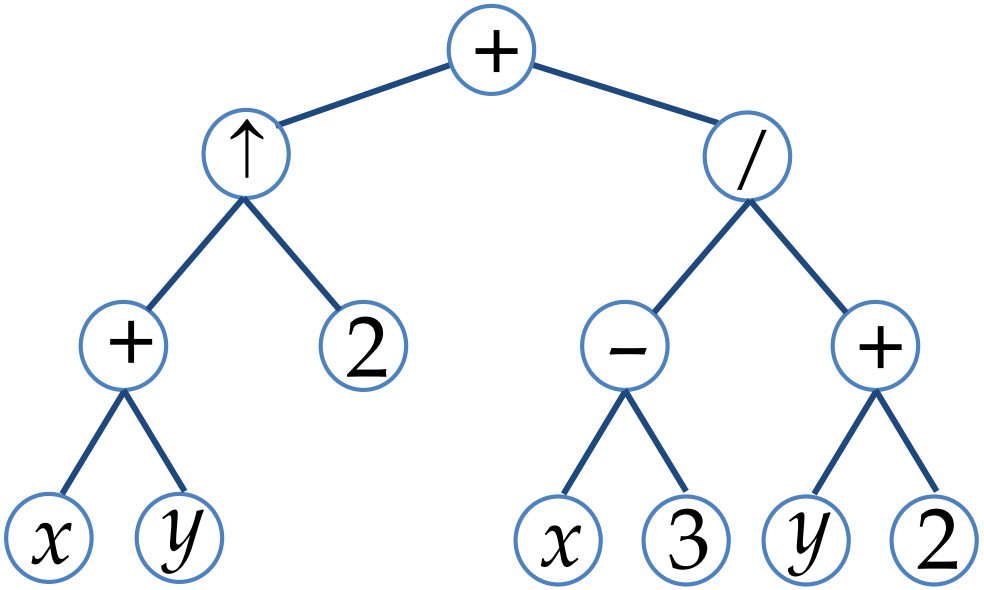
- In an prefix expression, a binary operator precedes its two operands
- The expression is evaluated right-left
- Look for the first operator from the right
- Evaluate the operator with the two operands immediately to its right

Example

+ / + 2 2 2 / – 3 2 + 1 0

Postfix Notation (Reverse Polish)

- Traverse in postorder



Evaluating Postfix Notation

- In an postfix expression, a binary operator follows its two operands
- The expression is evaluated left-right
- Look for the first operator from the left
- Evaluate the operator with the two operands immediately to its left

Example

2 2 + 2 / 3 2 – 1 0 + / +

Tree Traversal Algorithms

A Binary Expression Tree is . . .

A special kind of binary tree in which:

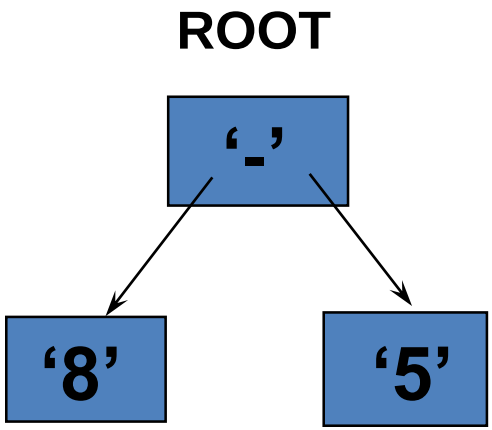
1. Each leaf node contains a single operand,

2. Each nonleaf node contains a single binary operator, and

3. The left and right subtrees of an operator node represent subexpressions that must be evaluated before applying the operator at the root of the subtree.

Tree Traversal Algorithms

A Binary Expression Tree



INORDER TRAVERSAL :

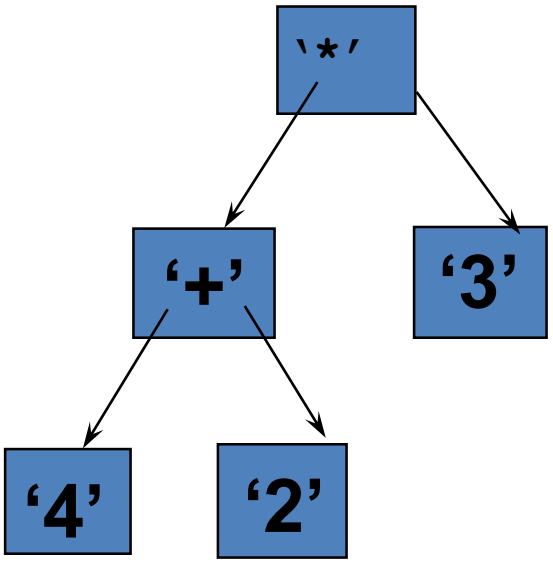
PREORDER TRAVERSAL:

POSTORDER TRAVERSAL:

UTD

Tree Traversal Algorithms

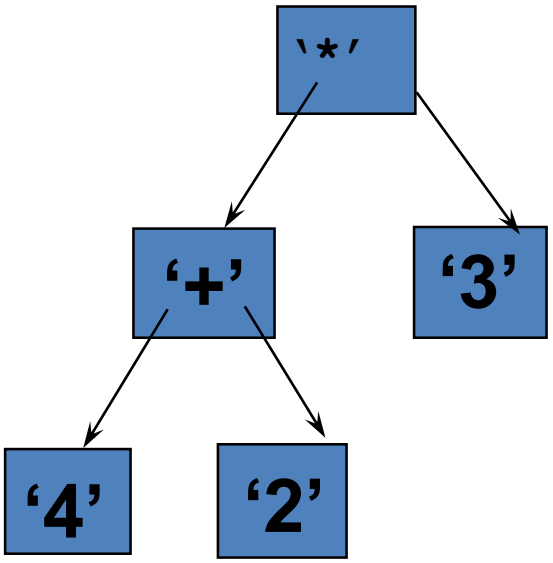
A Binary Expression Tree



What value does it have?

Tree Traversal Algorithms

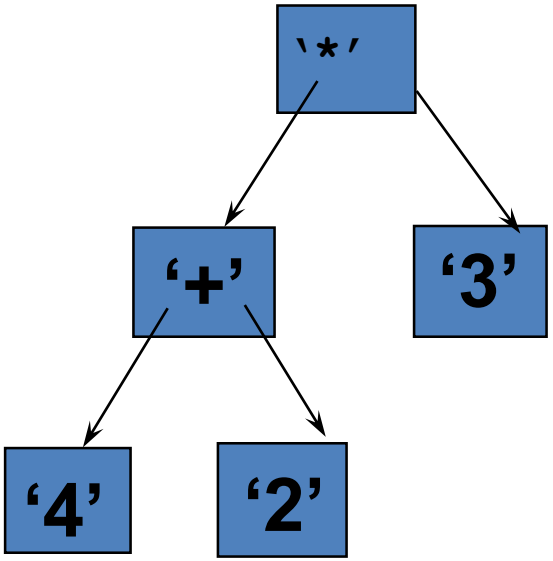
A Binary Expression Tree



What infix, prefix, postfix expressions does it represent?

Tree Traversal Algorithms

A Binary Expression Tree



Infix:

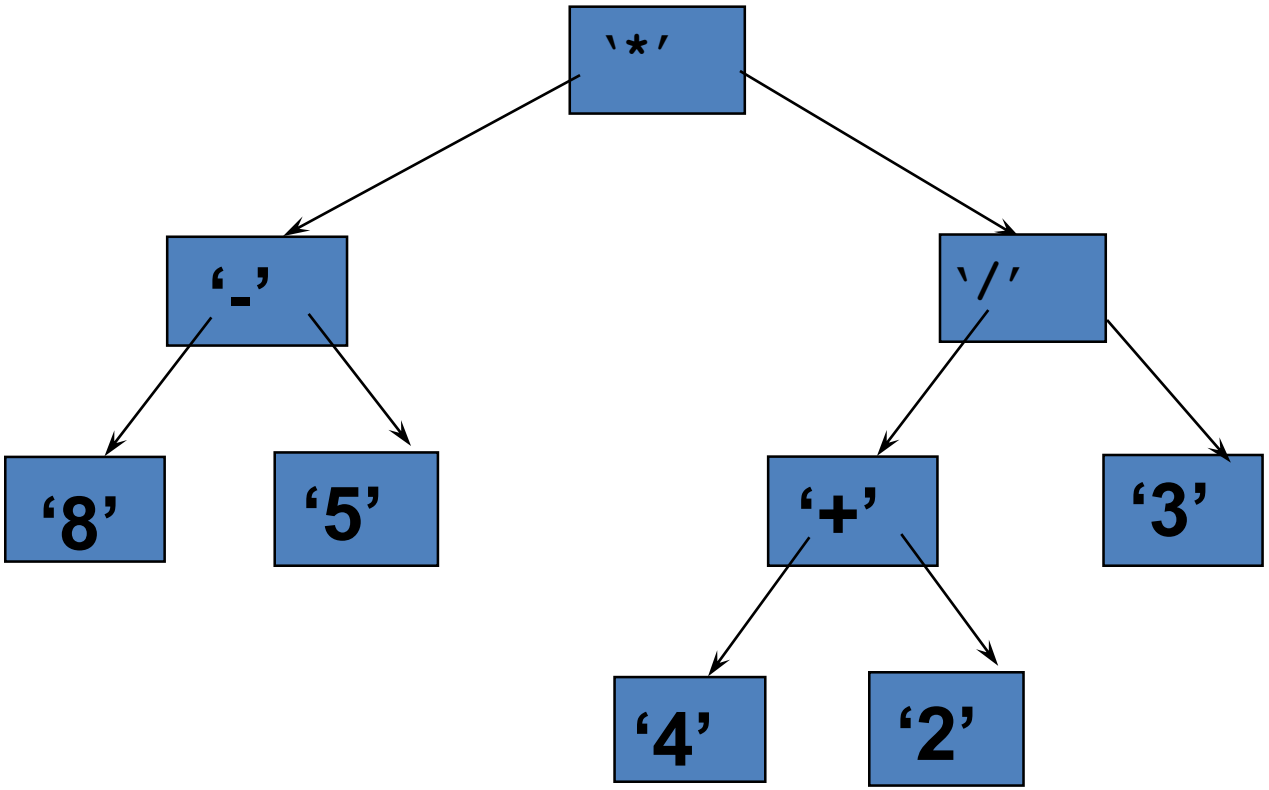
Prefix:

Postfix:

UTD

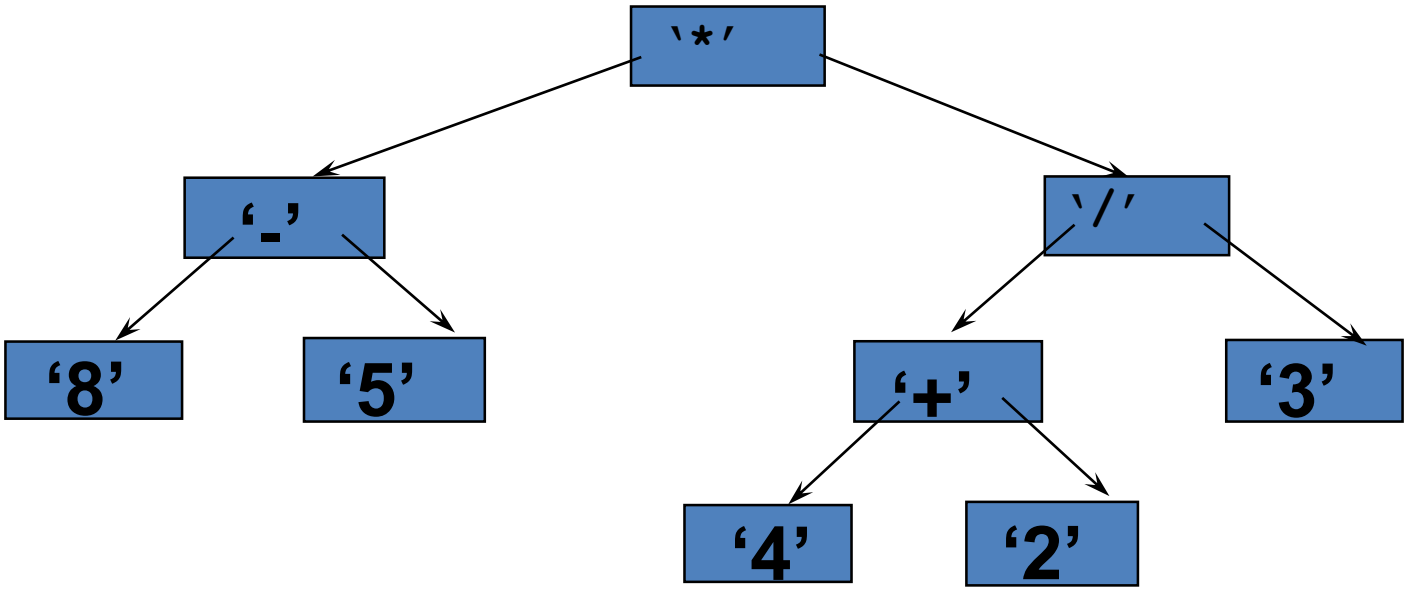
Tree Traversal Algorithms

Evaluate this binary expression tree



What infix, prefix, postfix expressions does it represent?

Tree Traversal Algorithms



Infix:

Prefix:

Postfix:

Applications

Where are preorder, inorder, and postorder traversals used?

Preorder –

Inorder –

Postorder –