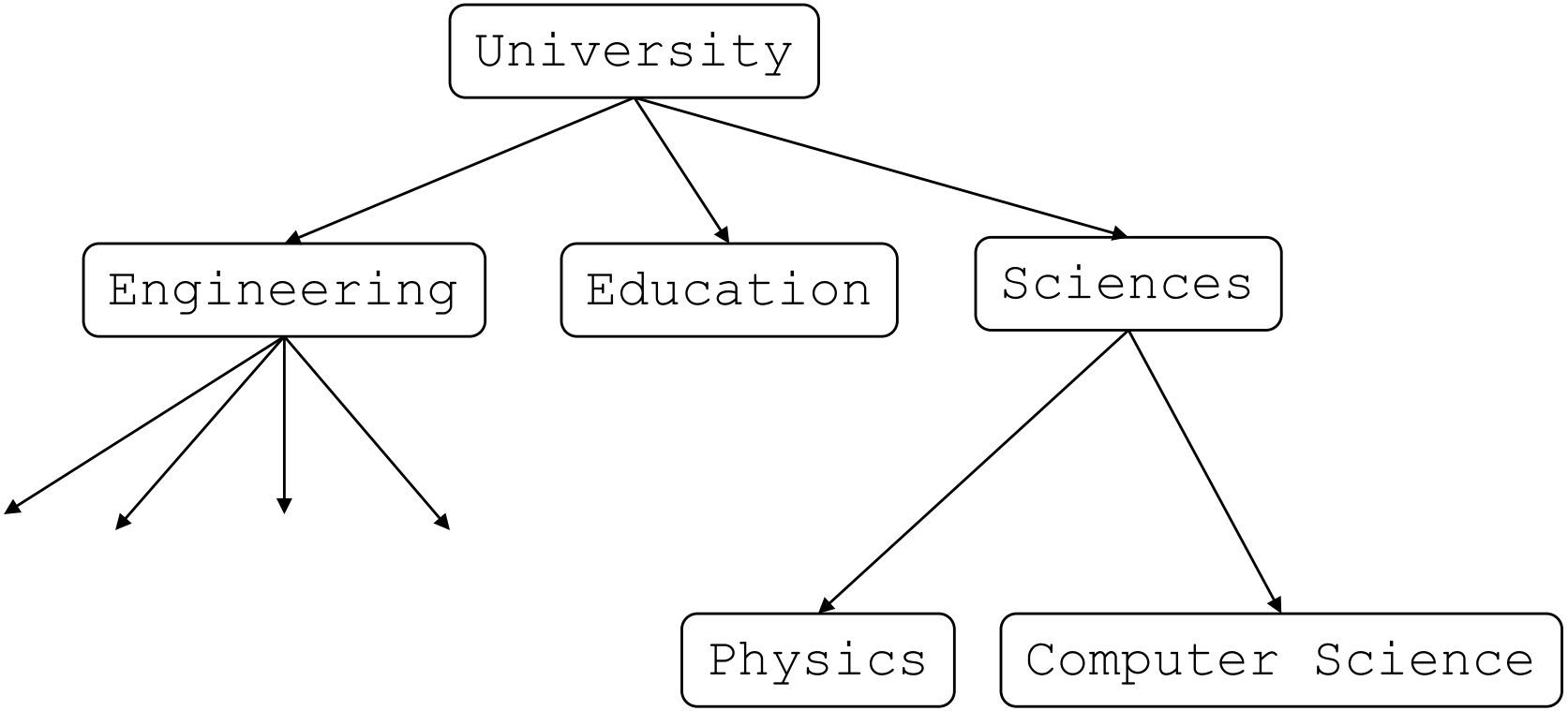
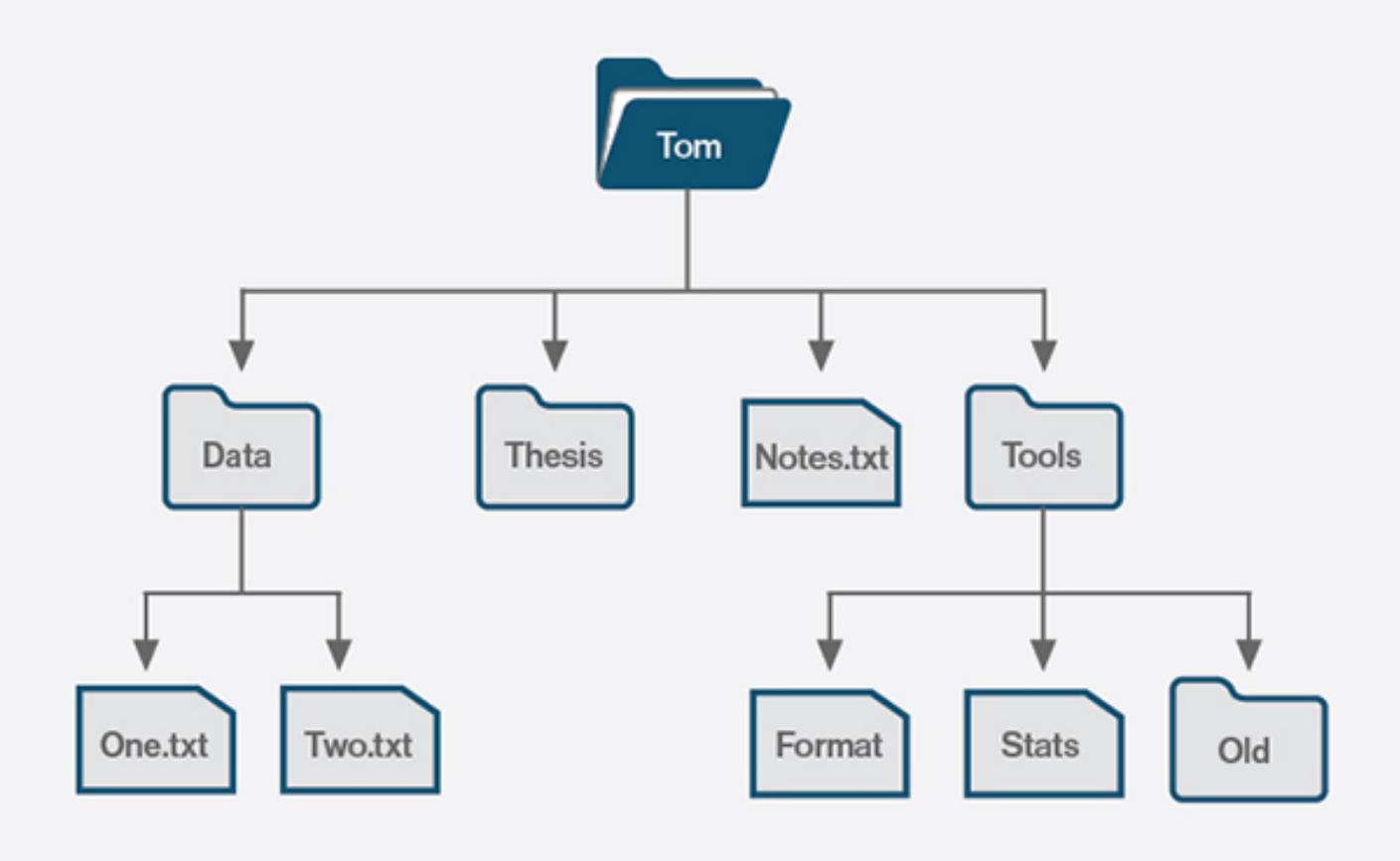
CMSC 132: Object-Oriented Programming II

Binary Trees

Trees

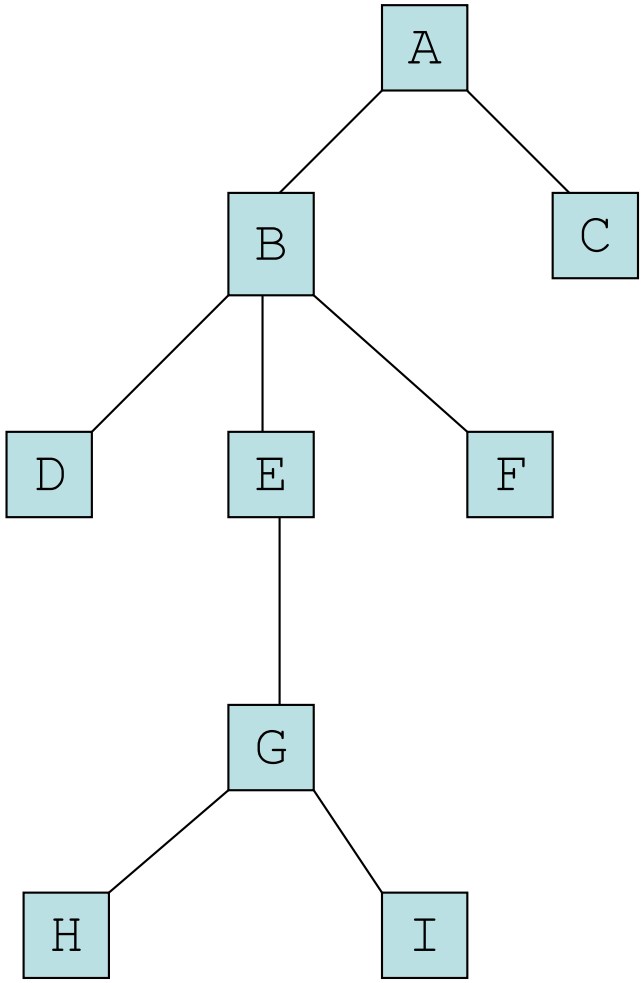

University
Engineering
Education
Sciences
Physics
Computer Science

Trees


Tom
Data
Thesis
Notes.txt
Tools
One.txt
Two.txt
Format
Stats
Old

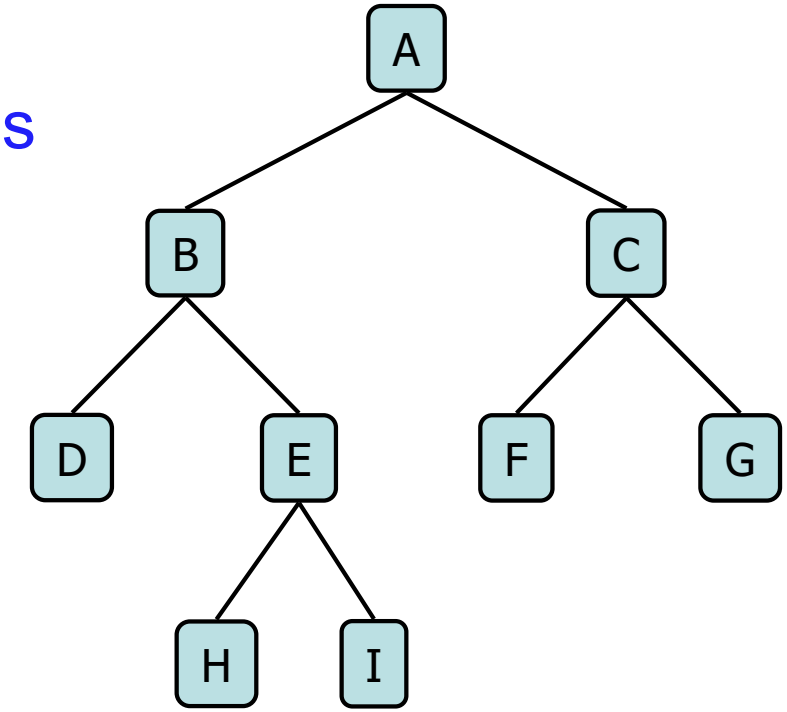
Trees

- A tree is a node with a value and zero or more children.
- No Cycle
- **Properties**
  - Number of nodes
  - Height
  - Root Node
  - Leaves
  - Interior nodes
  - Ancestor
  - Descendant
  - Siblings
  - Subtrees



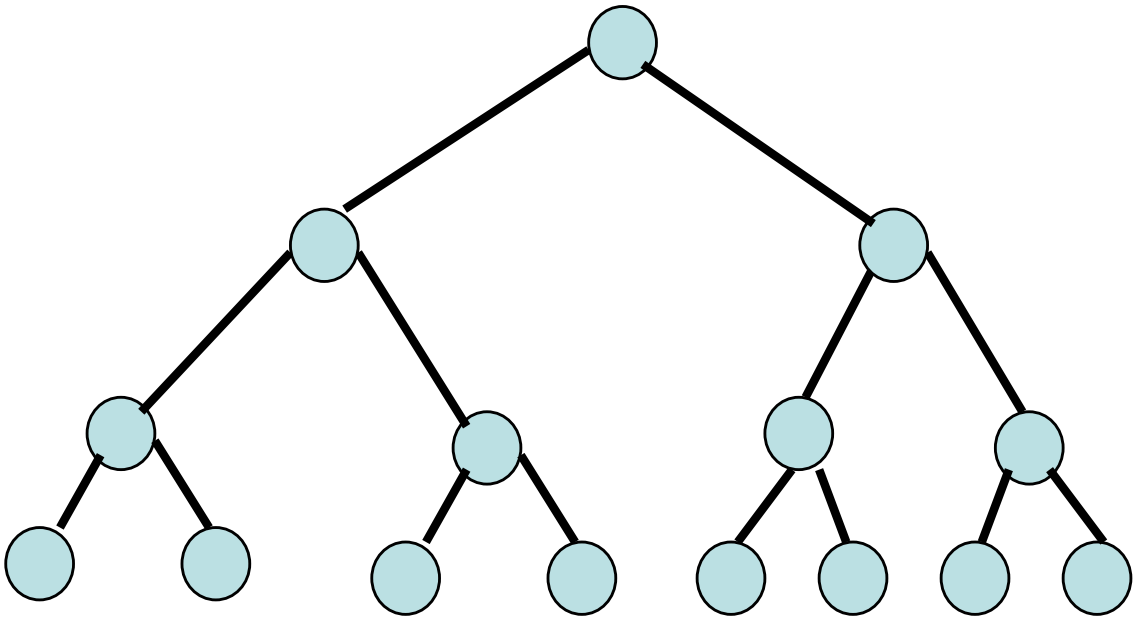
Binary Tree

- Each internal node has at most two children (degree of two)
- The children of a node are an ordered pair
- We call the children of an internal node left child and right child
- Applications:
  - arithmetic expressions
  - decision processes
  - searching



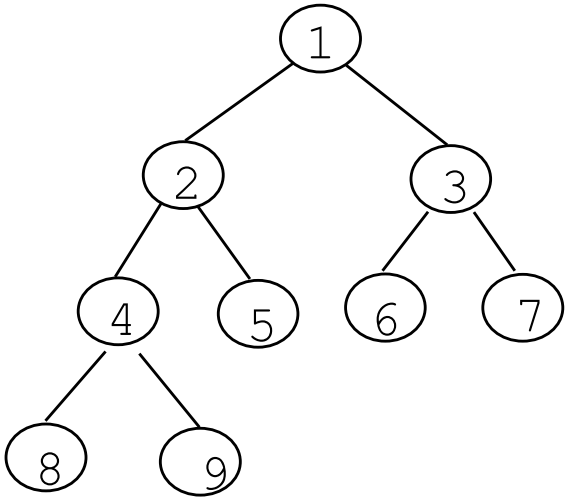
Full Binary Tree

- A full binary tree is a tree in which every node other than the leaves has two children.
- A full (perfect) binary tree of a given height k has $2^{k+1}-1$ nodes.

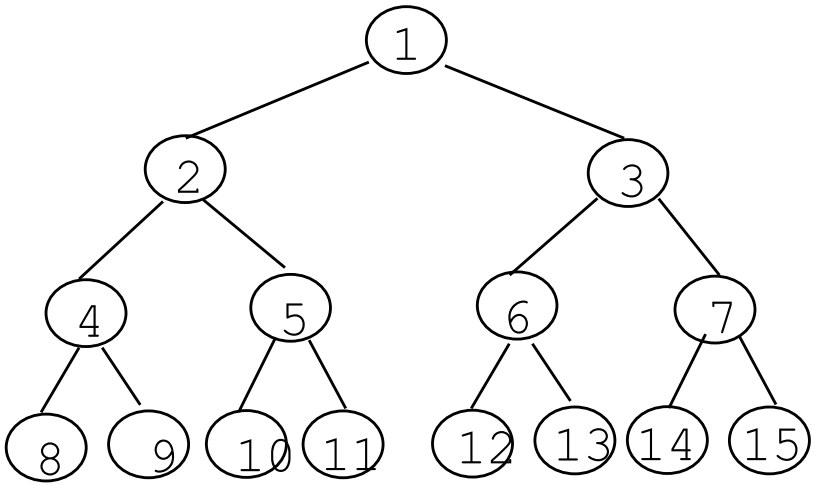


Complete Binary Trees

A **complete binary tree** is a **binary tree** in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

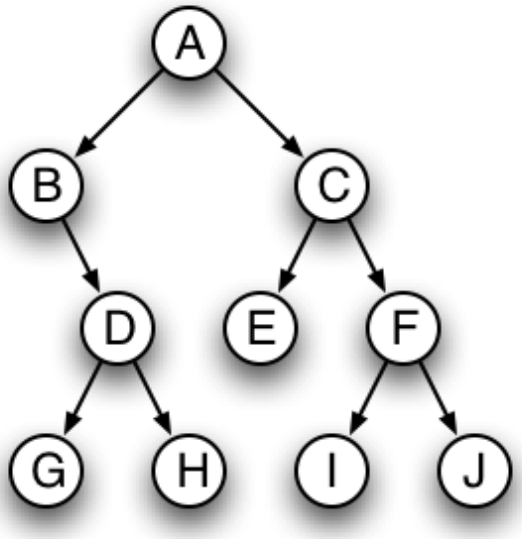


Complete binary tree



Full binary tree

Binary Tree Traversal



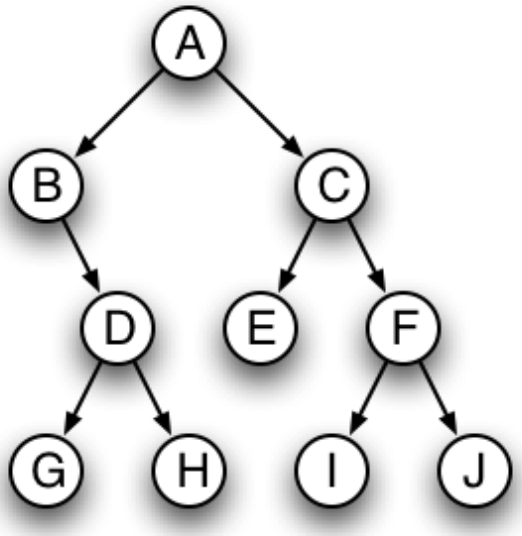
preOrder:

inOrder:

postOrder:

Level Order:

Binary Tree Traversal



preOrder: root, left, right
A B D G H C E F I J

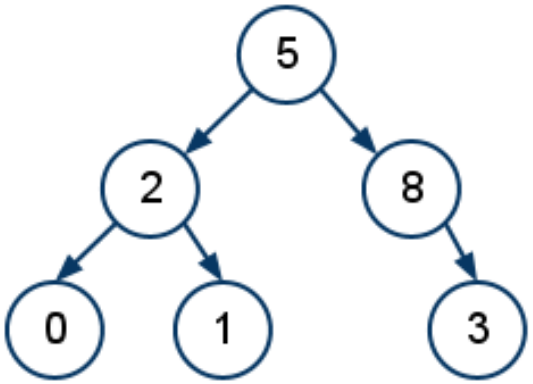
inOrder:left, root, right
B G D H A E C I F J

postOrder:left, right, root
G H D B E I J F C A

Level Order: BFS

Quiz 1:

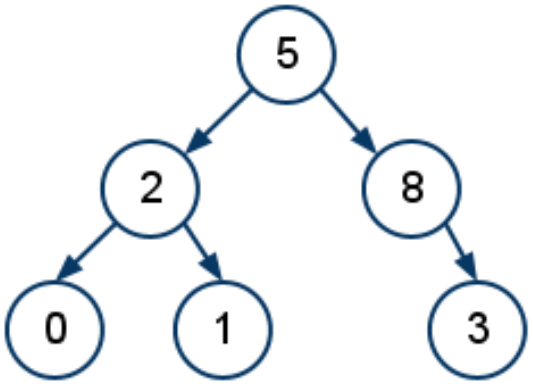
What is the preOrder traversal of this binary tree?



A. 5 2 8 0 1 3
B. 5 2 1 0 3 8
C. 5 2 0 1 8 3
D. 5 2 0 1 3 8

Quiz 1:

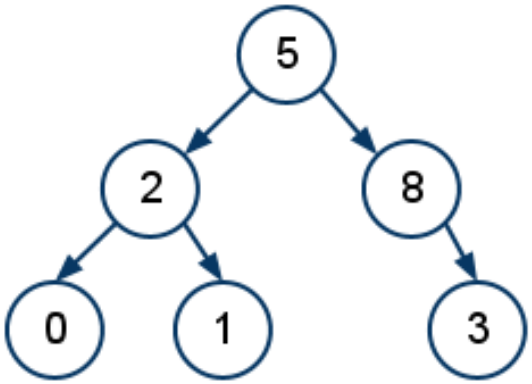
What is the preOrder traversal of this binary tree?



A. 5 2 8 0 1 3
B. 5 2 1 0 3 8
C. 5 2 0 1 8 3
D. 5 2 0 1 3 8

Quiz 2:

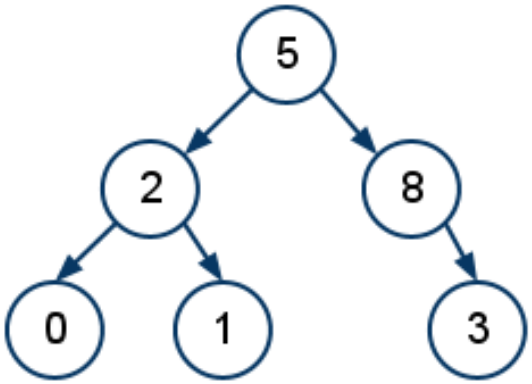
What is the inOrder traversal of this binary tree?



A. 0 1 2 3 8 5
B. 0 2 1 5 8 3
C. 0 2 1 5 3 8
D. 5 2 0 1 3 8

Quiz 2:

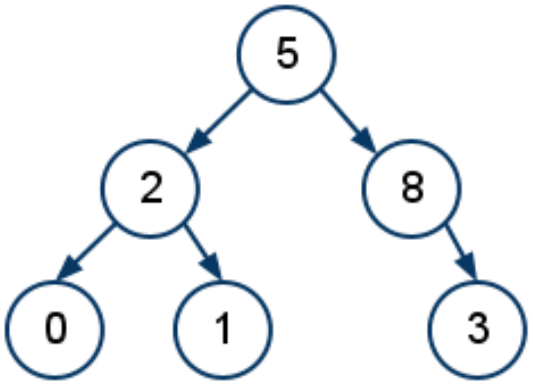
What is the inOrder traversal of this binary tree?



A. 0 1 2 3 8 5
B. 0 2 1 5 8 3
C. 0 2 1 5 3 8
D. 5 2 0 1 3 8

Quiz 3:

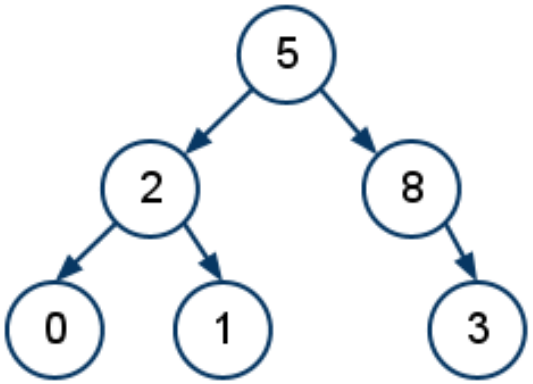
What is the postOrder traversal of this binary tree?



A. 0 1 2 3 8 5
B. 0 2 1 5 8 3
C. 0 1 2 5 3 8
D. 5 2 0 1 3 8

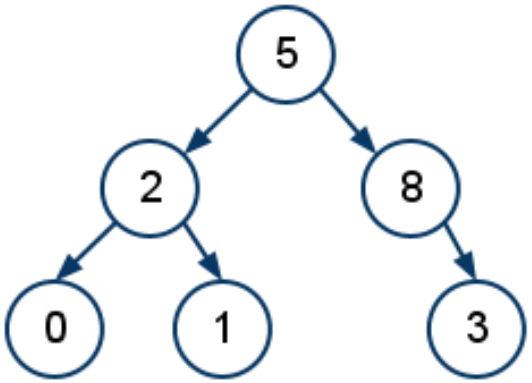
Quiz 3:

What is the postOrder traversal of this binary tree?



A. 0 1 2 3 8 5
B. 0 2 1 5 8 3
C. 0 1 2 5 3 8
D. 5 2 0 1 3 8

Binary Tree Traversal



preOrder: **5 2 0 1 8 3**

inOrder: **0 2 1 5 8 3**

postOrder: **0 1 2 3 8 5**

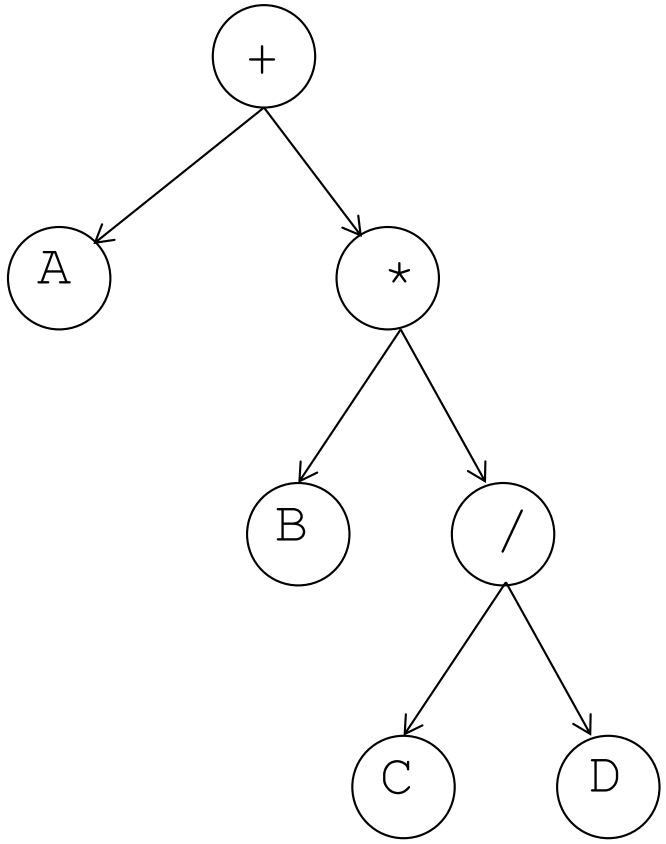
Level Order: 5 2 8 0 1 3

Arithmetic Expression Trees

Arithmetic Expression:

A + (B * (C / D))

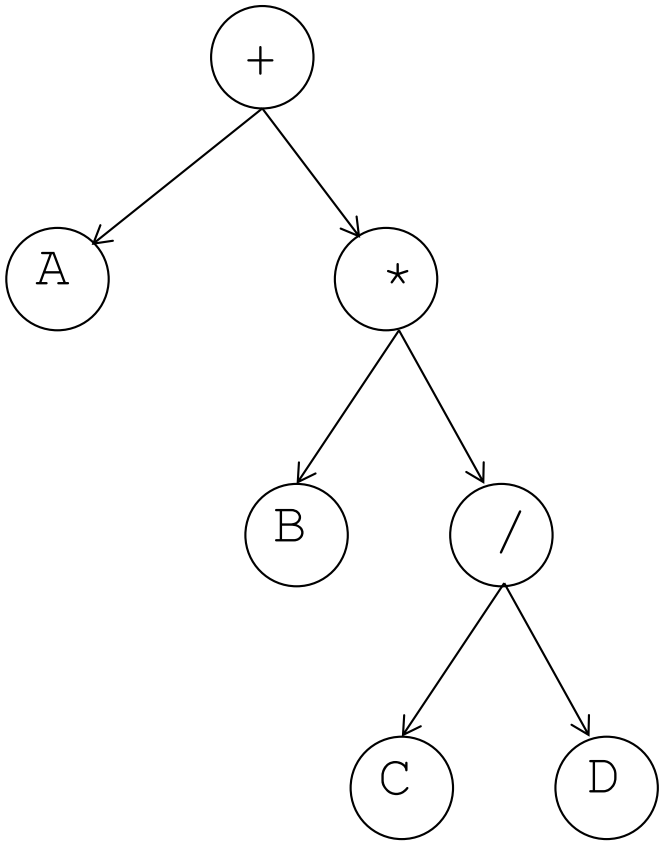
Tree for the above expression:



- Used in most compilers
- No parenthesis need to evaluate
- Calculate by traversing tree

Traversing Trees

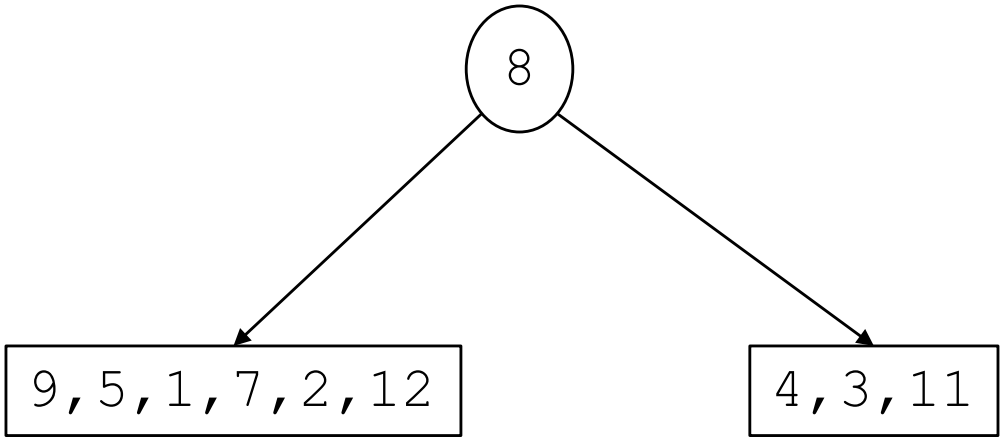
- Preorder: Root, then Children
  - + A * B / C D
- Postorder: Children, then Root
  - A B C D / * +
- Inorder: Left child, Root, Right child
  - A + B * C / D



Build a Binary Tree

Build a Binary Tree from given inOrder, postOrder

inOrder: 9,5,1,7,2,12,**8**,4,3,11
postOrder: 9,1,2,12,7,5,3,11,4,8 ← root

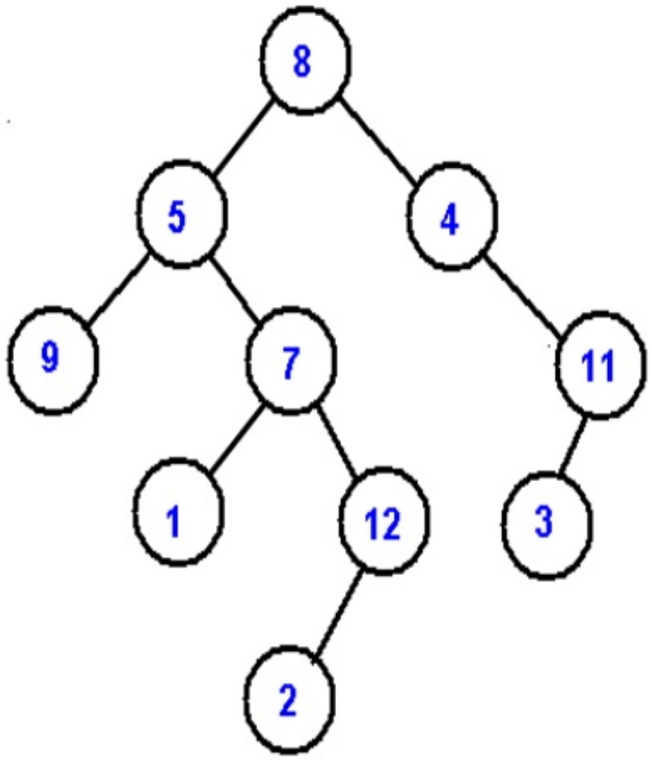


Build a Binary Tree

Build a Binary Tree from given inOrder, postOrder

inOrder: 9,5,1,7,2,12,8,4,3,11
postOrder: 9,1,2,12,7,5,3,11,4,8

PreOrder - 8, 5, 9, 7, 1, 12, 2, 4, 11, 3
InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
LevelOrder - 8, 5, 4, 9, 7, 11, 1, 12, 3, 2



Build a Binary

Build Binary Tree from inOrder, preOrder

Inorder : DBHEIAFCG

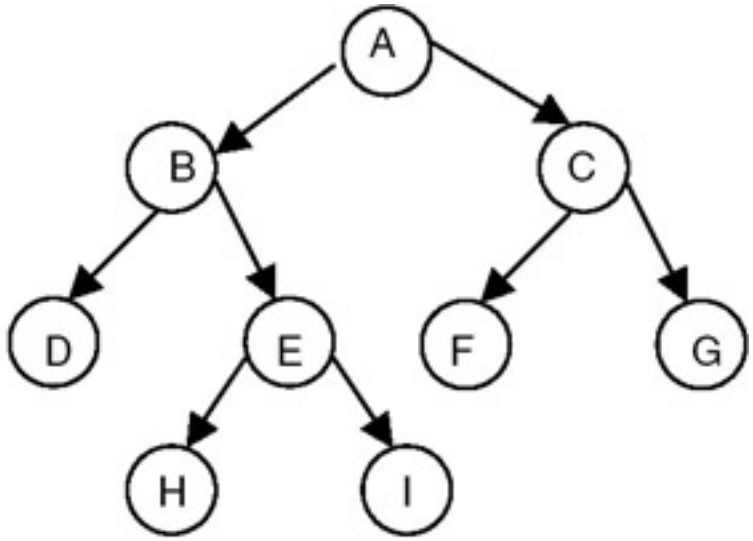
Preorder : ABDEHICFG

Build a Binary

Build Binary Tree from inOrder, preOrder

Inorder : DBHEIAFCG

Preorder : ABDEHICFG



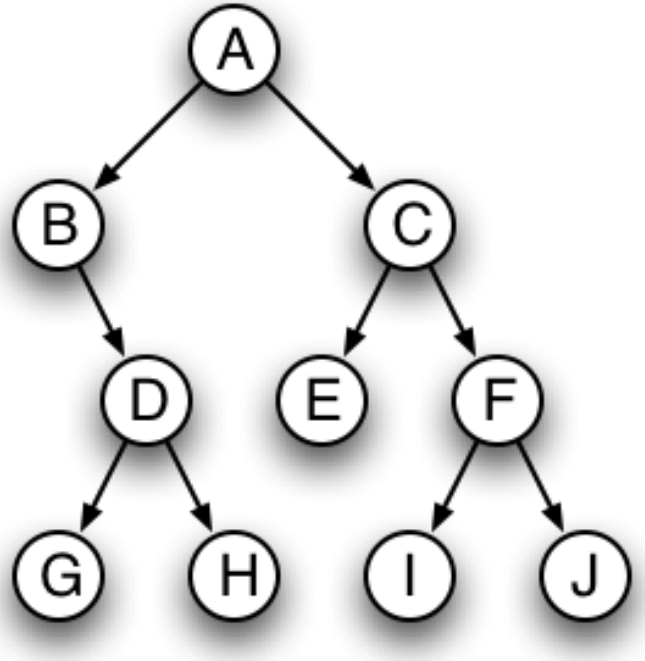
Inorder : DBHEIAFCG

Preorder : ABDEHICFG

Postorder : DHIEBFGCA

Binary Tree Implementation

Height:
Size:
Diameter:
Mirror:
Path:
Least Common Ancestor (LCA):



Binary Tree Node Class

```
class Node {
   private E key;
   private Node left, right;
   Node(E key) {
      this.key = key;
  }
}
```

Binary Tree Class

```
public class BinaryTree<E> {
       private Node root;
       class Node {
         private E key;
         private Node left, right;
         Node(E key) {
            this.key = key;
         }
       }
}
```

Binary Tree Implementation

Check out the Binary Tree code examples from github