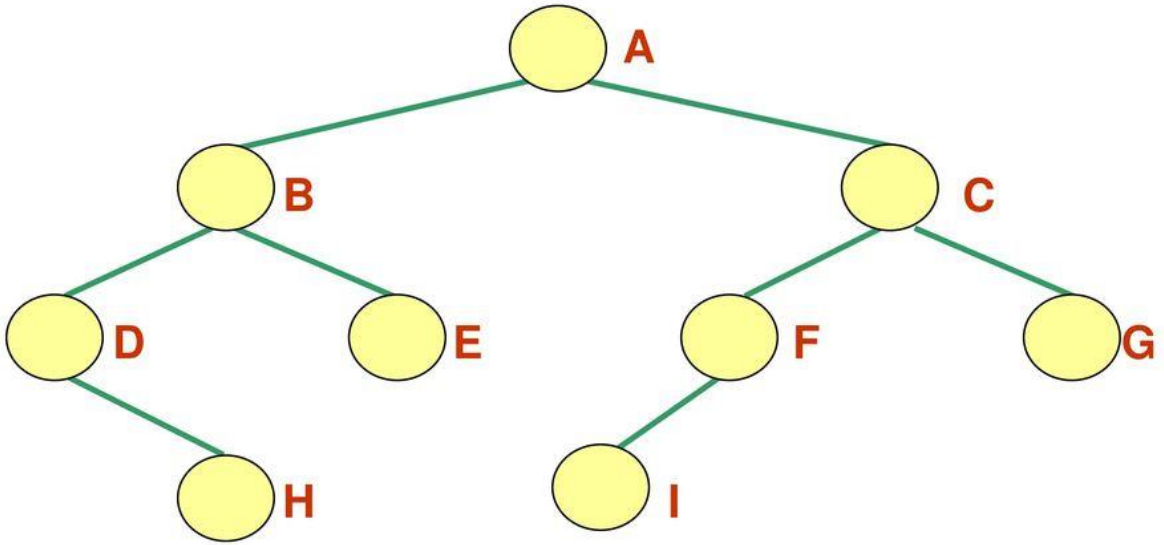
Tree Traversal

Tree Traversals

- A ***traversal*** of a tree requires that each node of the tree be ***visited*** once
  - ***Example***: a typical reason to traverse a tree is to display the data stored at each node of the tree
- Standard traversal orderings:
  - ***preorder***
  - ***inorder***
  - ***postorder***
  - ***level-order***

10-22

Traversals



We'll trace the different traversals using this tree; recursive calls, returns, and "visits" will be numbered in the order they occur

10-23

Preorder Traversal

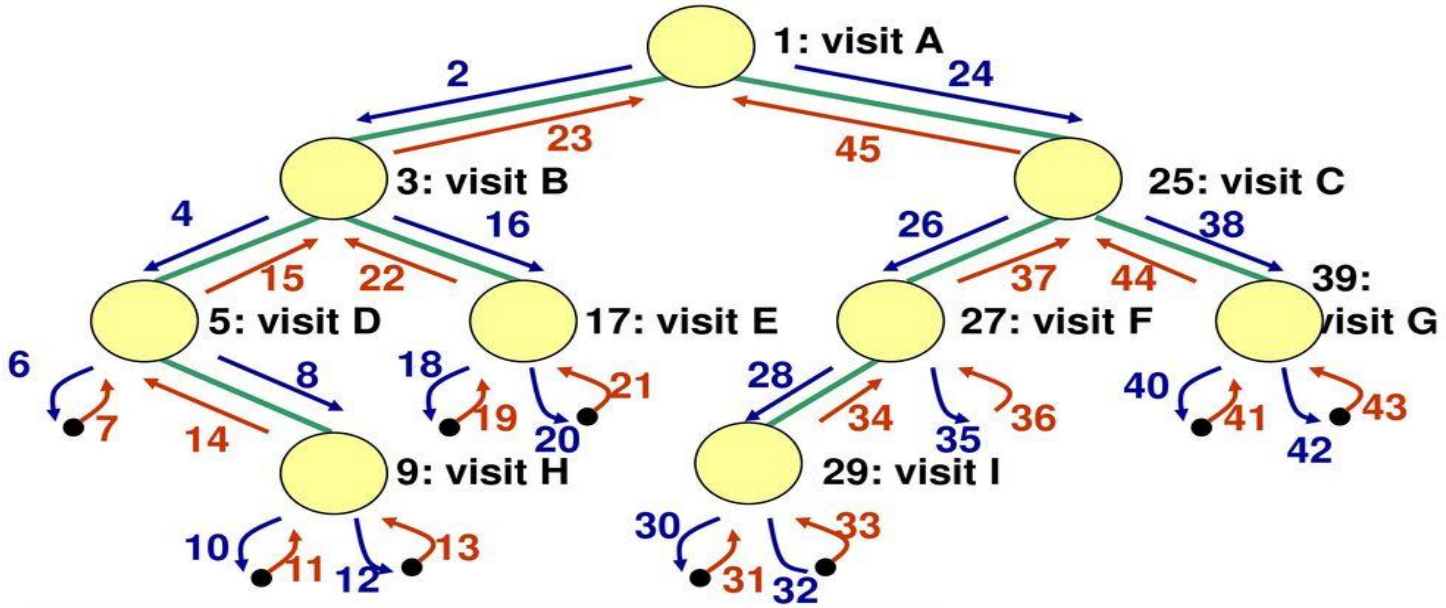
- Start at the root
- Visit each node, followed by its children; we will choose to visit left child before right

- ***Recursive algorithm*** for preorder traversal:
  - If tree is not empty,
    - Visit root node of tree
    - Perform preorder traversal of its left subtree
    - Perform preorder traversal of its right subtree

  - *What is the base case?*
  - *What is the recursive part?*

10-24

Preorder Traversal



Nodes are visited in the order ABDHECFIG

10-25

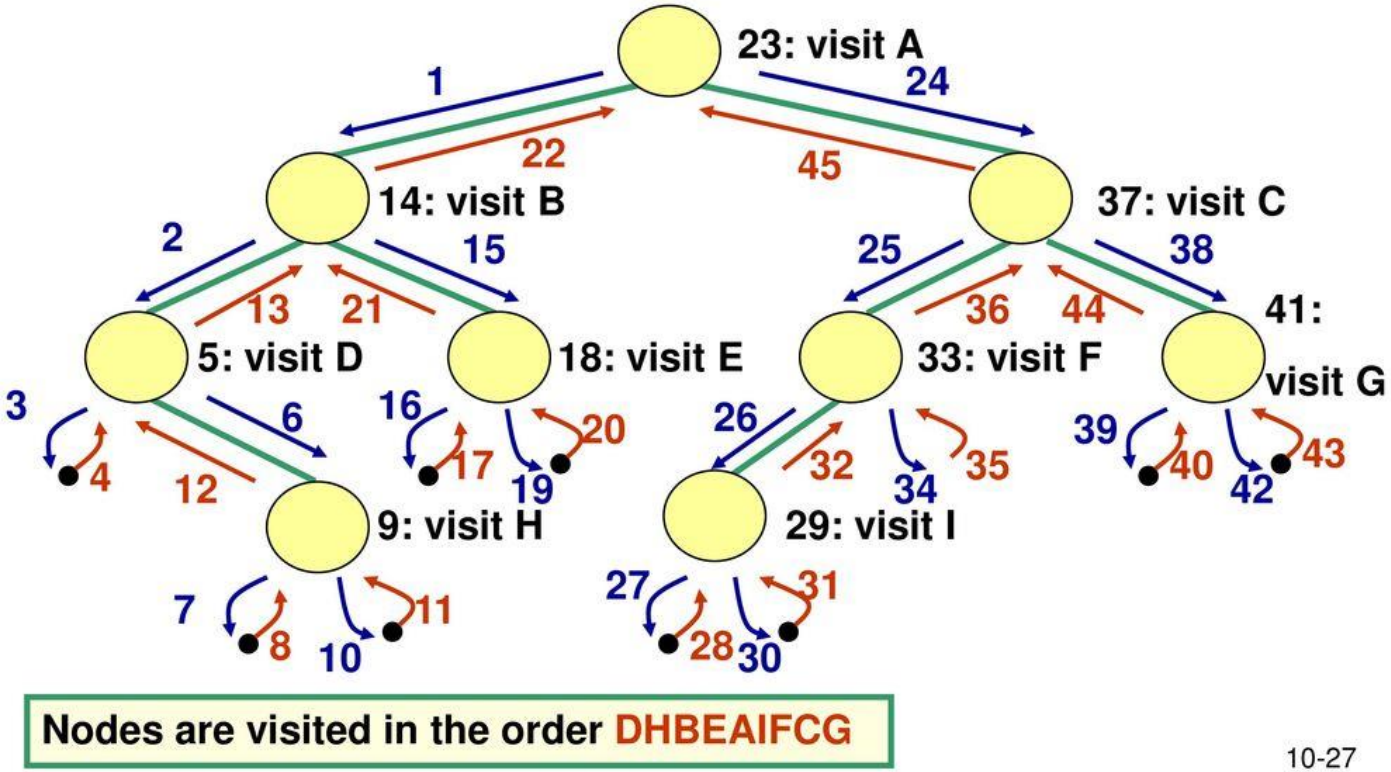
Inorder Traversal

- Start at the root
- Visit the left child of each node, then the node, then any remaining nodes

- ***Recursive algorithm*** for inorder traversal
  - If tree is not empty,
    - Perform inorder traversal of left subtree of root
    - Visit root node of tree
    - Perform inorder traversal of its right subtree

10-26

Inorder Traversal



Nodes are visited in the order DHBEAIFCG

10-27

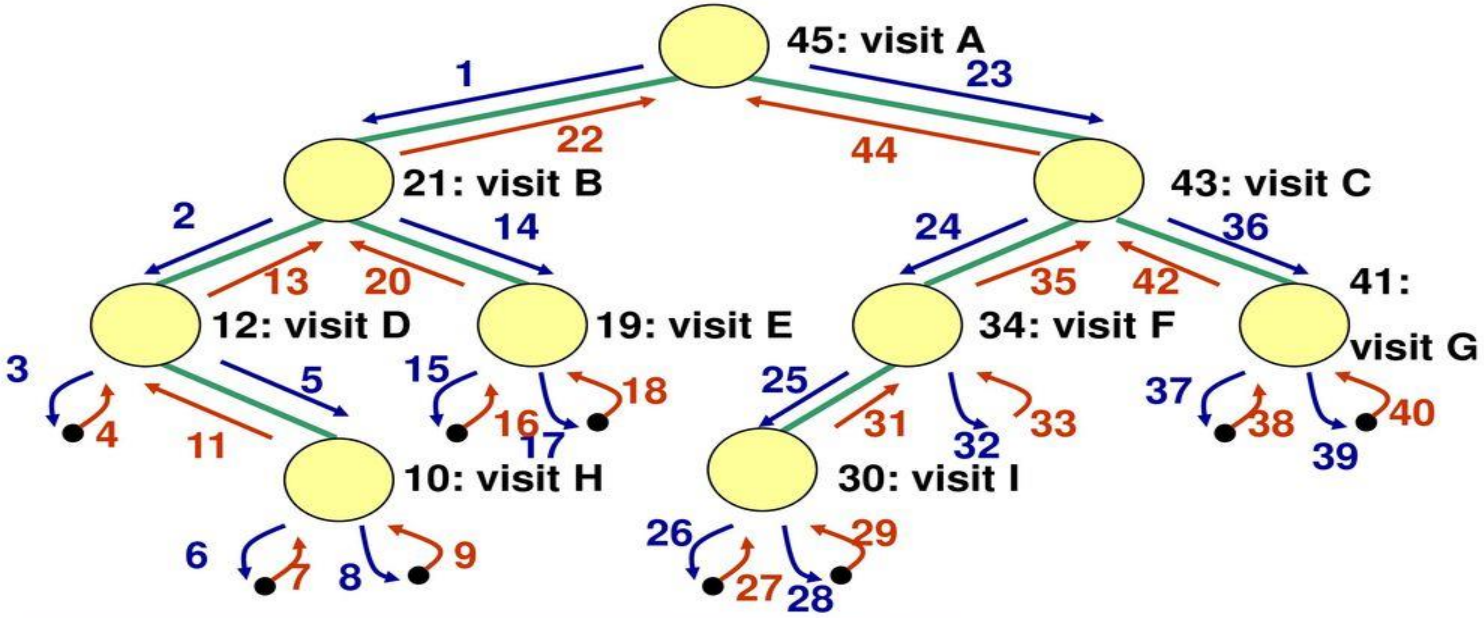
Postorder Traversal

- Start at the root
- Visit the children of each node, then the node

- ***Recursive algorithm*** for postorder traversal
  - If tree is not empty,
    - Perform postorder traversal of left subtree of root
    - Perform postorder traversal of right subtree of root
    - Visit root node of tree

10-28

Postorder Traversal



Nodes are visited in the order HDEBIFGCA

10-29