Trees

CMSC 132

Department of Computer Science

University of Maryland, College Park

Overview

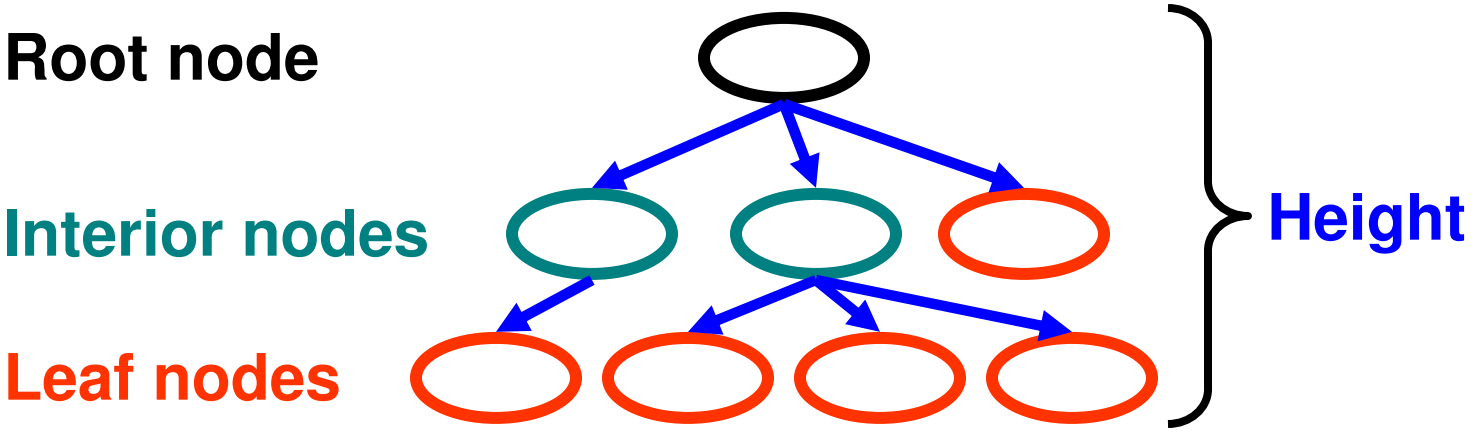
- Tree terminology
- Binary trees
- Types of binary trees
- Binary trees properties
- General trees

Tree Terminology

- A tree consists of a collection of elements or nodes
- Each node can contain data and (parent and child) links to other nodes
- Each node has one parent, except for the root node, which has no parent
- Each node can have any number of children

Tree Terminology

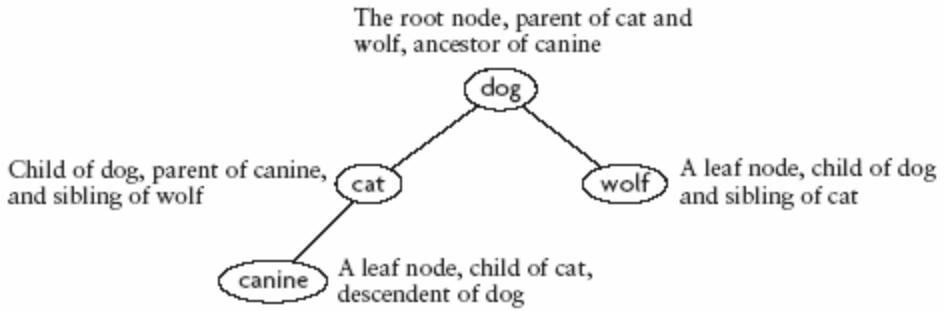
- **Terminology**
  - **Root ⇒ no predecessor**
  - **Leaf ⇒ no successor**
  - **Interior ⇒ non-leaf**
  - **Height ⇒ distance from root to leaf**



Tree Terminology

- Nodes that have the same parent are siblings
- A node that has no (non-empty) children is called a leaf node
- A generalization of the parent-child relationship is the ancestor-descendent relationship

FIGURE 8.2
A Tree of Words



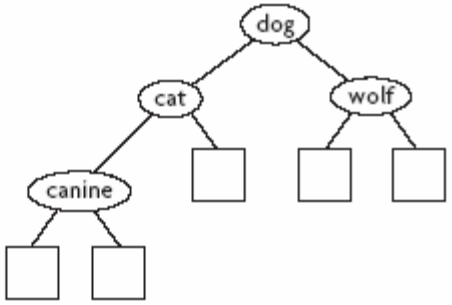
Tree Terminology

- A subtree of a node is a tree whose root is a child of that node
- The level of a node is a measure of its distance from the root.
- Definition of level
  - If node x is the root of the tree, its level is 1
  - If node x is not the root, its level is 1 + parent's level
- The height of a tree is the maximum level of any node in the tree
  - The empty tree has height zero
  - You will also hear depth used interchangeably for height

Binary Trees

- In a binary tree, each node has at most two subtrees
- A set of nodes T is a binary tree if either of the following is true
  - T is empty
  - Its root node has two subtrees, TL and TR, such that TL and TR are binary trees

FIGURE 8.3
A Tree of Words with Null Subtrees Indicated



Binary Tree Implementation

- **Node implementation**

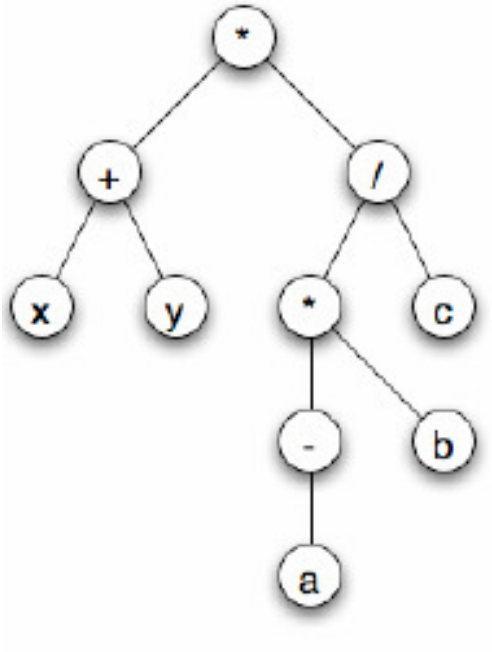
```
Class Node {
    Value data;
    Node left, right;     // null if empty
        ...
}
```

- **Later on we will see a polymorphic tree implementation**

Some Types of Binary Trees

- **Expression tree**
  - Each node contains an operator or an operand
- **Huffman tree**
  - Represents Huffman codes for characters that might appear in a text file
  - Huffman code uses different numbers of bits to encode letters as opposed to ASCII or Unicode
- **Binary search trees**
  - All elements in the left subtree precede those in the right subtree

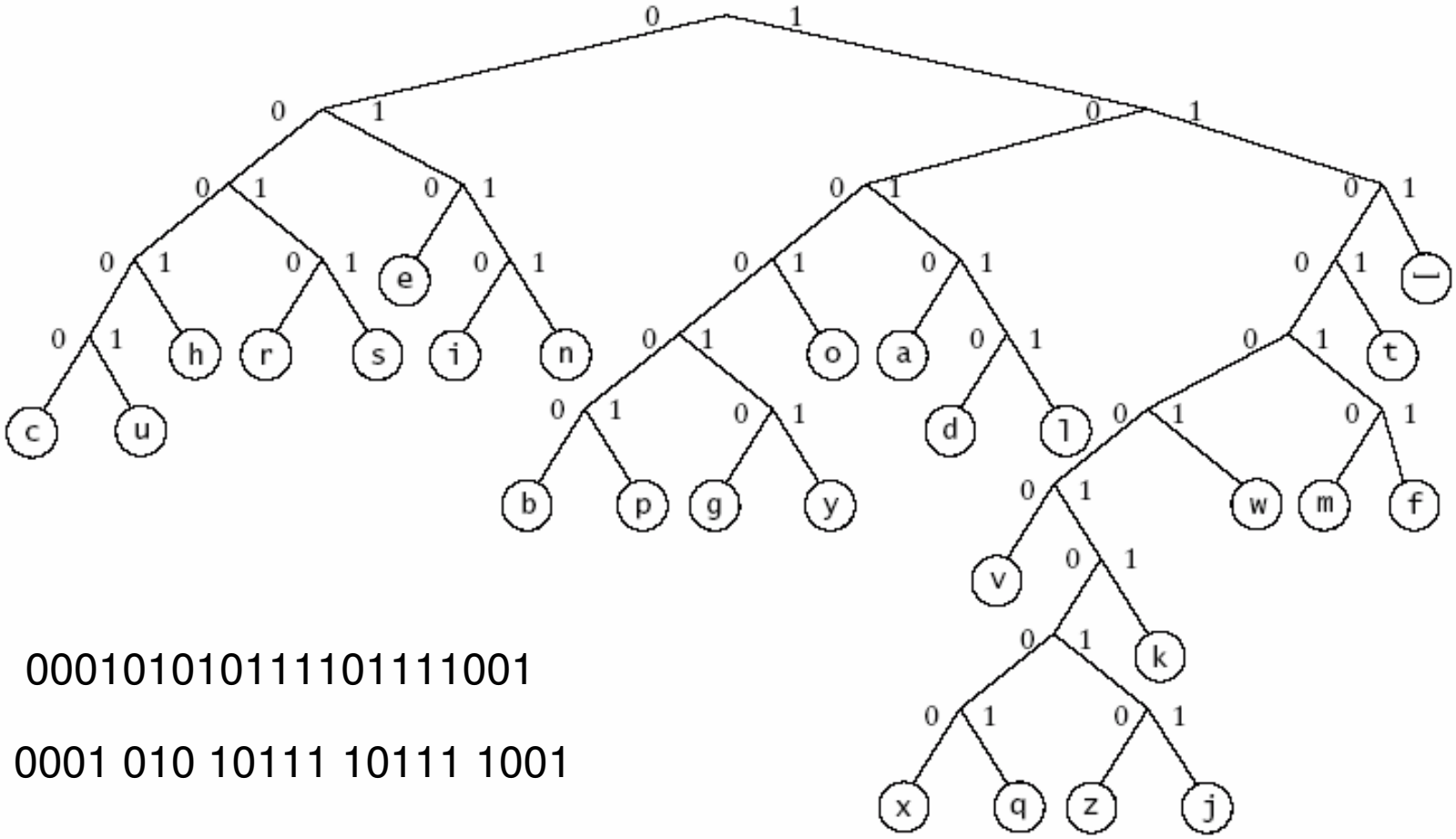
Expression tree



Huffman Trees

FIGURE 8.35

Huffman Tree Based on Frequency of Letters in English Text



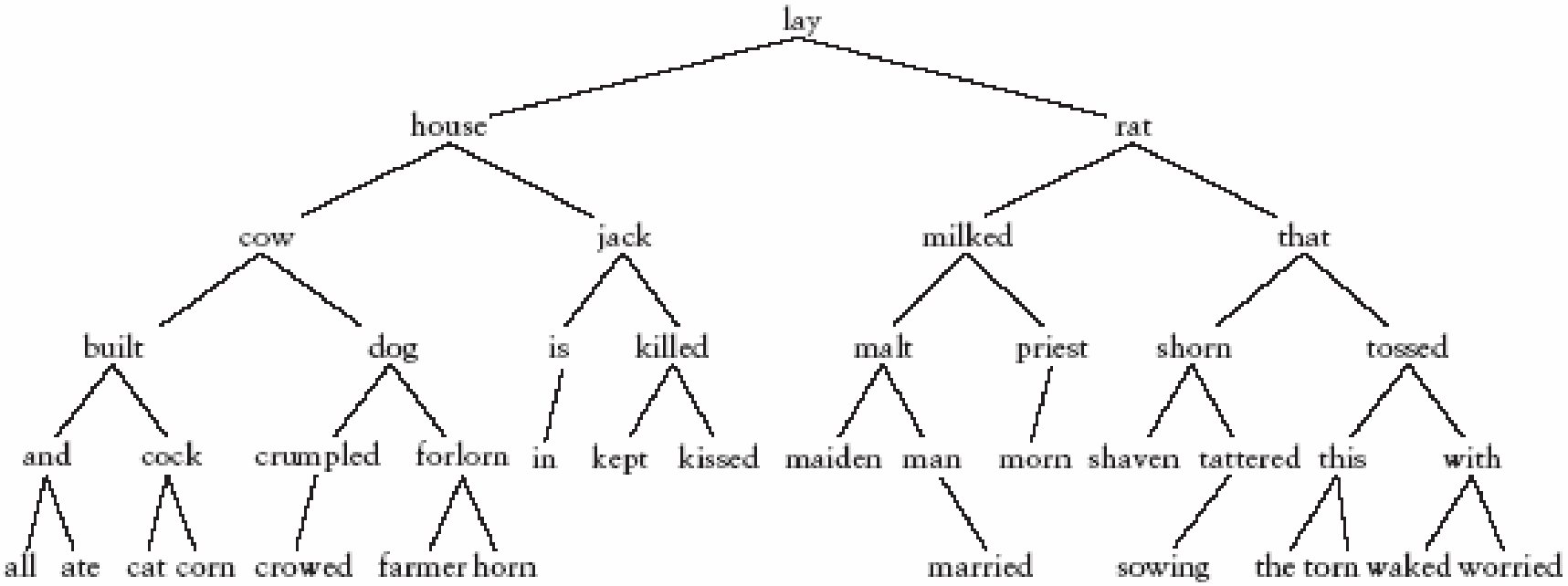
00010101011110111100 1

0001 010 10111 10111 1001

Binary Search Tree

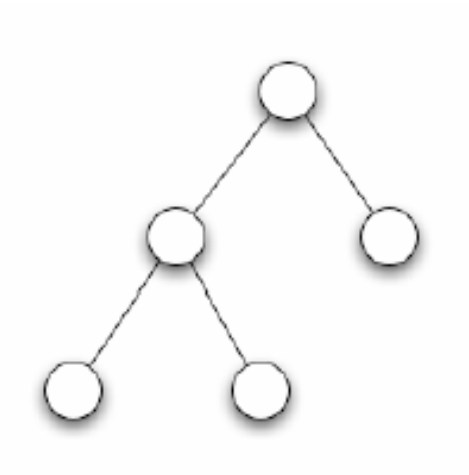
FIGURE 8.13

Binary Search Tree Containing All of the Words from "The House That Jack Built"



Full Binary Trees

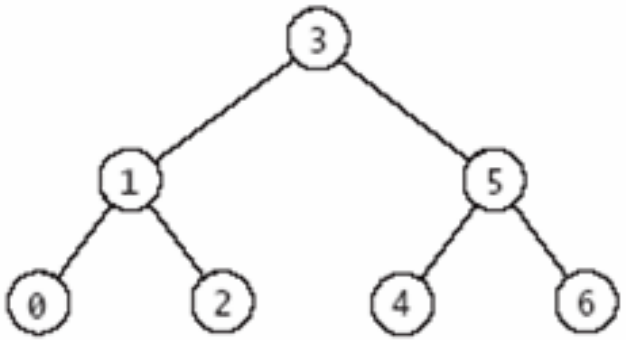
- A binary tree in which each node has exactly zero or two children (every node is either a leaf or has two children)



- From now on assume that whenever we refer to a node that has k children, we mean k non-empty children
- Warning: Book doesn't actually give a definition of full binary trees

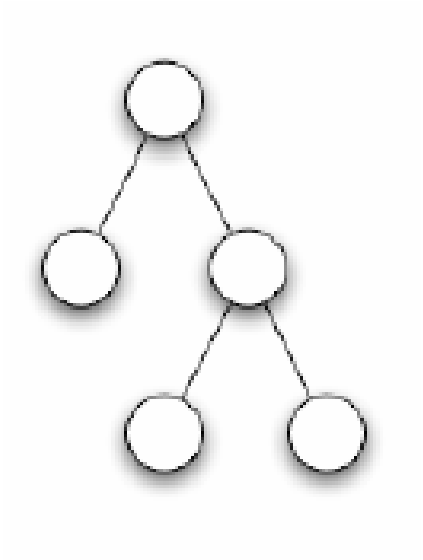
Perfect Binary Trees

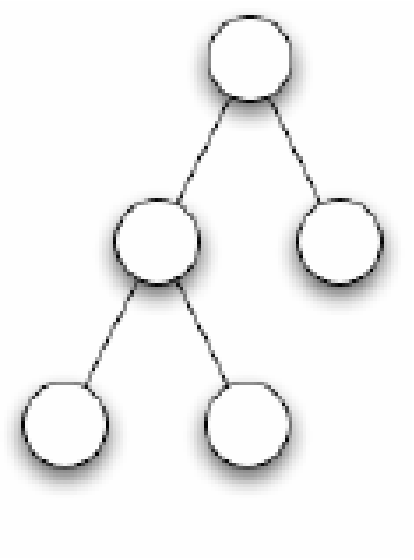
- Definition 1: A binary with all the leaf nodes at the same depth
- Definition 2: A binary tree is perfect it is a full binary tree and all leaf nodes are at the same level
- Definition 3: Recursively, a binary tree is perfect if
  - it is empty, or
  - it has two perfect children with equal height

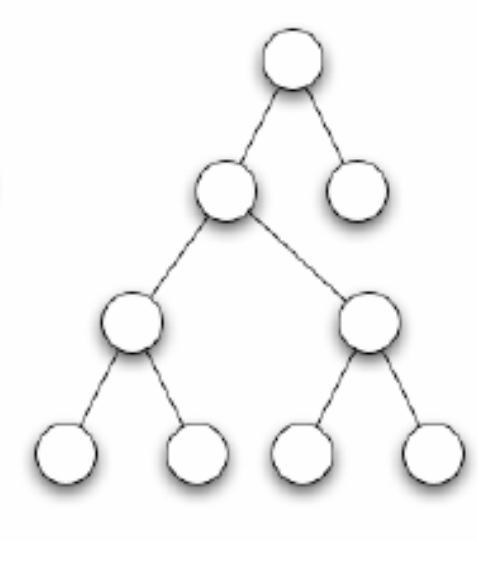


Complete Binary Trees

- **A binary tree of height h is complete if it is filled up at level h-1, and at height h, any unfilled nodes are on the right**





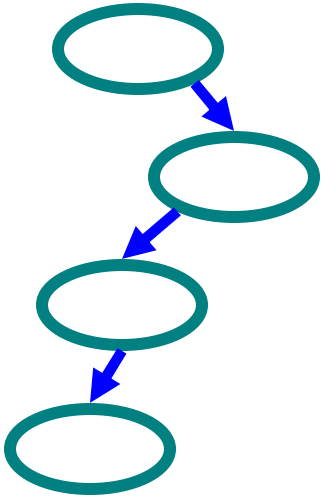


Nodes in a Complete Tree

- Nodes in a complete tree can be numbered
- Root is numbered 0
- Children of a node numbered x are numbered 2x+1 and 2x+2
- The nodes of a complete tree with n nodes are numbered 0..n-1

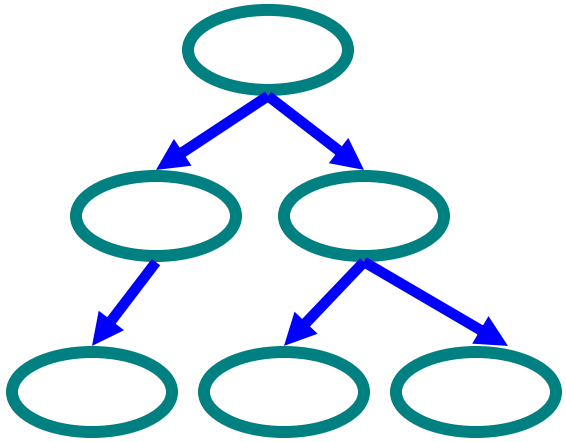
Binary Trees Properties

- **Degenerate**
  - Height = O(n) for n nodes
  - Similar to linear list



Degenerate binary tree

- **Balanced**
  - Height = O(log(n)) for n nodes
  - Useful for searches



Balanced binary tree

General Trees

- **Nodes of a general tree can have any number of subtrees**
- **A general tree can be represented using a binary tree**

FIGURE 8.8
Binary Tree Equivalent of King William's Family Tree

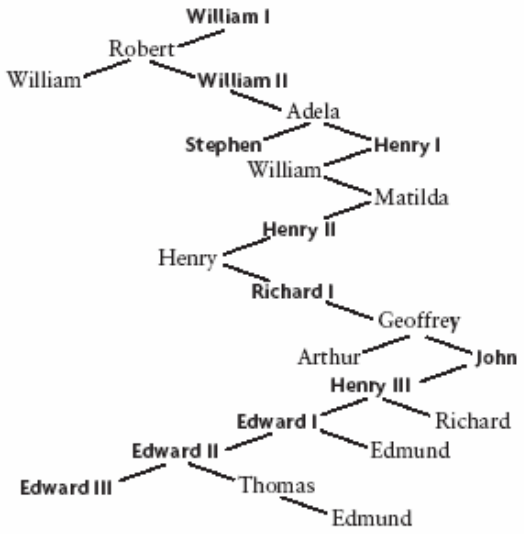
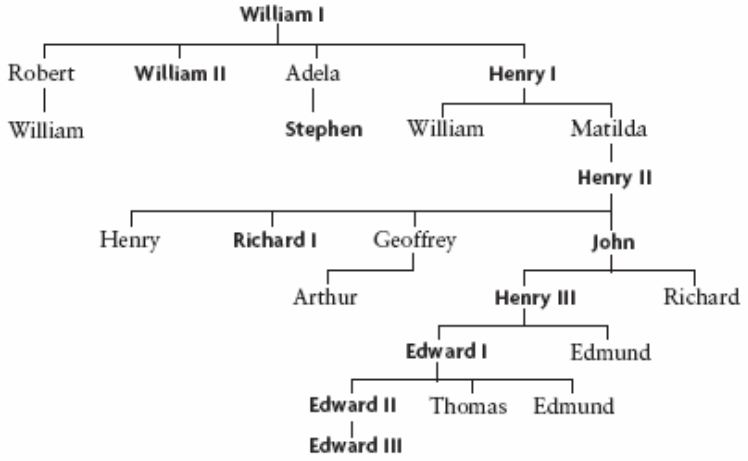


FIGURE 8.7
Family Tree for the Descendants of William I of England



Tree Traversals

- Often we want to determine the nodes of a tree and their relationship
  - Can do this by walking through the tree in a prescribed order and visiting the nodes as they are encountered
    - This process is called tree traversal
- Three kinds of tree traversal
  - Inorder
  - Preorder
  - Postorder

Tree Traversals

- **Preorder: Visit root node, traverse TL, traverse TR**
- **Inorder: Traverse TL, visit root node, traverse TR**
- **Postorder: Traverse TL, Traverse TR, visit root node**

Algorithm for Preorder Traversal

1. `if` the tree is empty
2. Return.

`else`

3. Visit the root.
4. Preorder traverse the left subtree.
5. Preorder traverse the right subtree.

Algorithm for Inorder Traversal

1. `if` the tree is empty
2. Return.

`else`

3. Inorder traverse the left subtree.
4. Visit the root.
5. Inorder traverse the right subtree.

Algorithm for Postorder Traversal

1. `if` the tree is empty
2. Return.

`else`

3. Postorder traverse the left subtree.
4. Postorder traverse the right subtree.
5. Visit the root.

Visualizing Tree Traversals

- You can visualize a tree traversal by imagining a mouse that walks along the edge of the tree
  - If the mouse always keeps the tree to the left, it will trace a route known as the Euler tour
    - Preorder traversal if we record each node as the mouse first encounters it
    - Inorder if each node is recorded as the mouse returns from traversing its left subtree
    - Postorder if we record each node as the mouse last encounters it

Visualizing Tree Traversals

FIGURE 8.9
Traversal of a Binary Tree