EEL 4837
Programming for Electrical Engineers II

Ivan Ruchkin
Assistant Professor
Department of Electrical and Computer Engineering
University of Florida at Gainesville
iruchkin@ece.ufl.edu
http://ivan.ece.ufl.edu

Trees

Readings:

- Weiss 4.1–4.3
- Horowitz 2.2, 6.1
- Cormen 10.4, 12

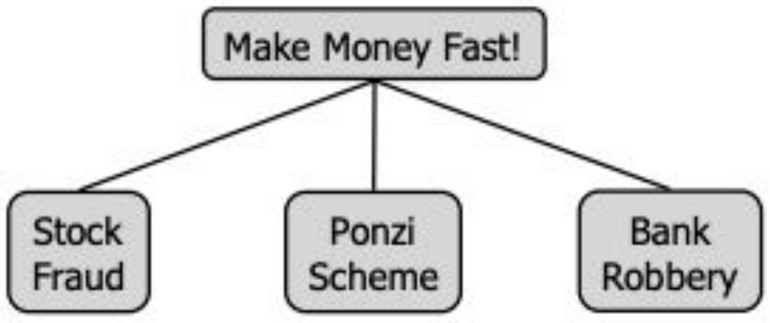
Trees

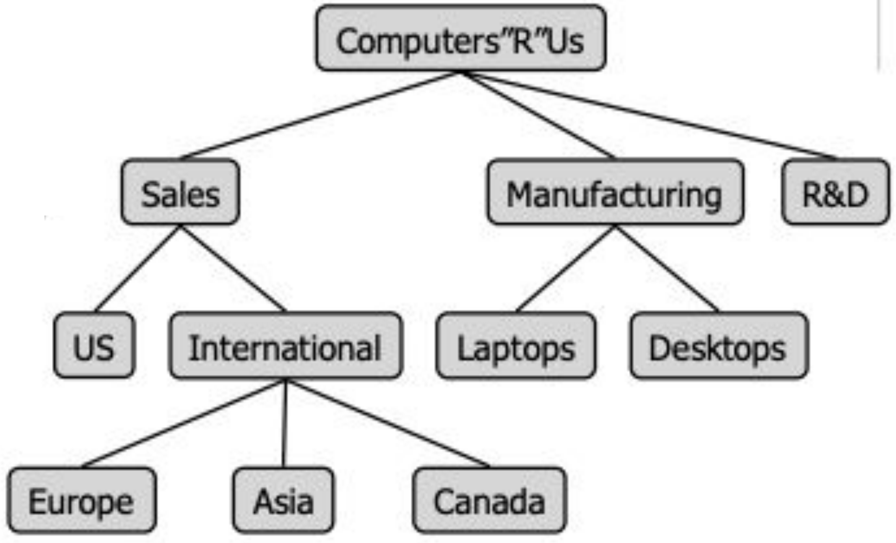
An abstract model of a hierarchical structure

- Consists of a collection of "nodes" related by a parent-child relation

Applications

- Organization charts
- File systems
- Programming environments



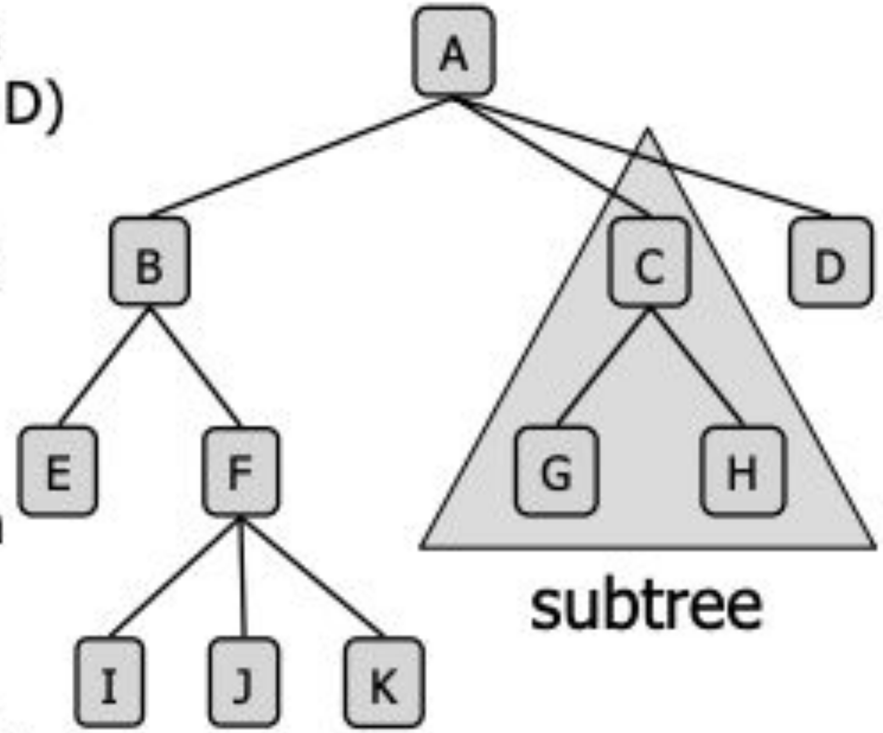


What Are Trees, Formally?

- Trees are defined *recursively*
- A *tree* is a **collection of nodes**. The collection can be empty; otherwise, a tree consists of a distinguished node r, called the **root**, and zero or more *non-empty disjoint* **(sub) trees** T_1, T_2, …, T_k, each of whose roots are connected by a directed edge from r
- So, a **tree** is a collection of a **root node** and the **trees** it connects to

Tree Terminology

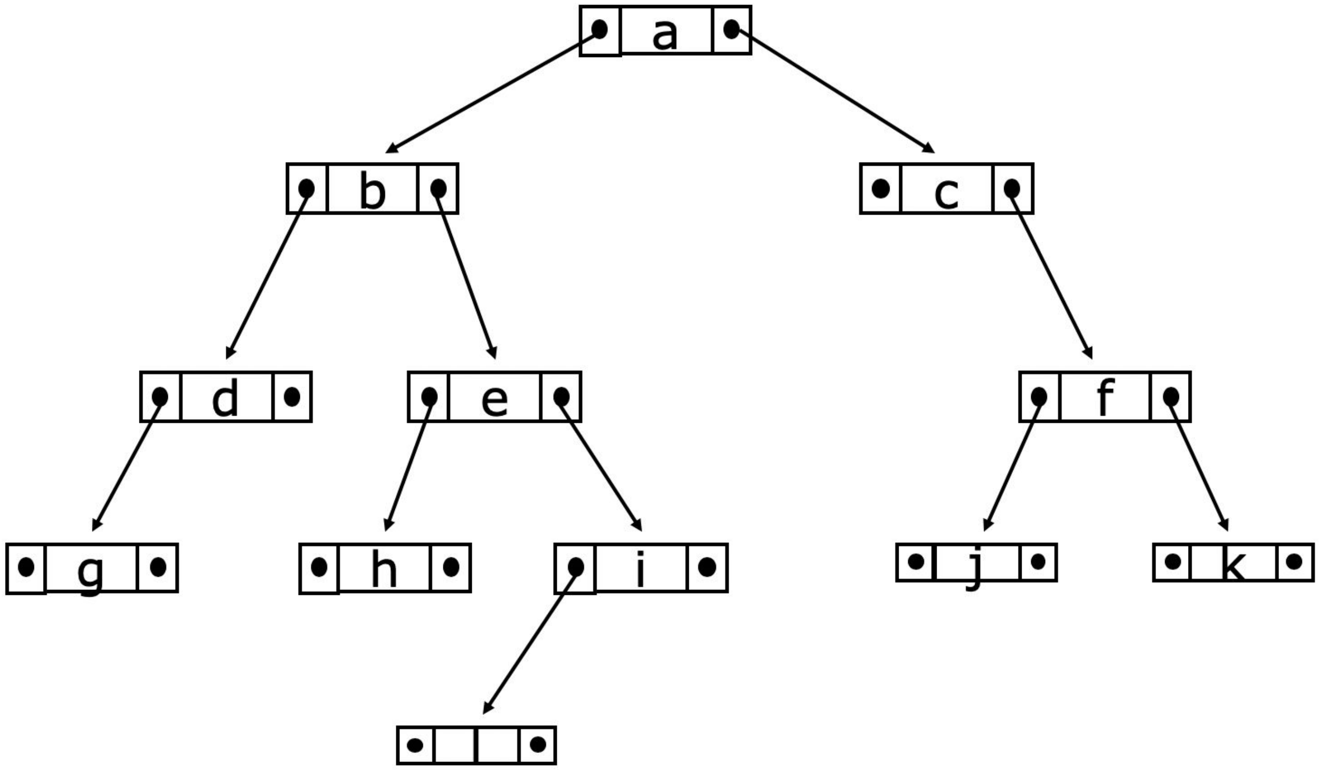
- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (a.k.a. leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Depth of a node: number of ancestors
- Height of a tree: maximum depth of any node (3)
- Descendant of a node: child, grandchild, grand-grandchild, etc.
- Subtree: tree consisting of a node and its descendants



Tree Terminology (Continued)

- A **path** from node n_1 to n_k is defined as a sequence of nodes n_1, n_2, …, n_k such that n_i is the parent of n_{i+1} for 1<= i <= k
- The **length** of this path is the **number of edges** on the path, namely k-1
- The length of the path *from a node to itself* is 0
- There is exactly one path from the root to each node

Binary Trees



- A **binary tree** is a tree in which **no** node can have **more than two children**
- In practice, each node has an **element** (“data”), a reference to a **left child** and a reference to a **right child**

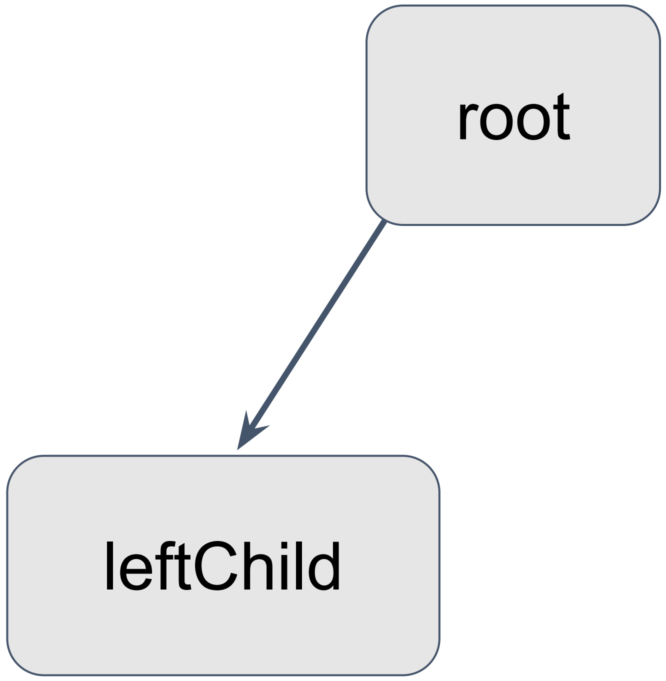
Questions for you:

- What is the minimum height of a binary tree with n nodes? Maximum?
- What is the maximum number of leaves in a binary tree of height h? Minimum?
- What is the maximum number of nodes in a binary tree of height h? Minimum?

Binary Tree in C++

```
template<typename T=int>
class Node {
public:
    T data; // the data element
    Node* left;
    Node* right;
    Node* parent;
}; // could define convenient constructors etc

// building a tree with two nodes
Node* root = new Node();
Node* leftChild = new Node();
root->left = leftChild;
leftChild->parent = root;
// don't forget to assign NULLs to other pointers
root->right = root->parent = NULL;
leftChild->right = leftChild->parent = NULL;
```



Tree Operations

We will write algorithms to:

- Compute the height (aka depth) of a binary tree
- Compute the number of nodes in a binary tree

Finding the Height of a Tree

```
int findMax(int a, int b){
      if(a >= b)
         return a;
      else
      return b;
}

int findHeight(Node* root) {
   // base case
   if(root == NULL)
         return 0;

   // recursive case
   return findMax( findHeight(root->left), findHeight(root->right) ) + 1;
}
```

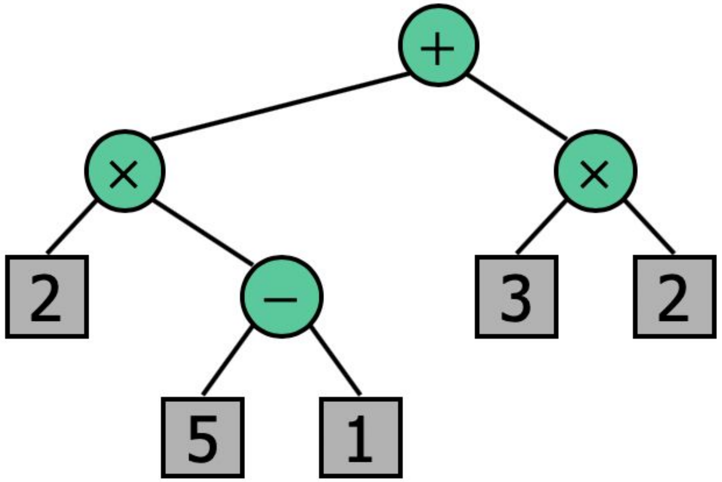
Finding the Number of Nodes of a Tree

```
int findNumber (Node* root){
    // base case
    if(root == NULL)
        return 0;

    // recursive case
    return findNumber(root->left) + findNumber(root->right) + 1;
}
```

Arithmetic Expression Tree

- A binary tree can store an arithmetic expression
  - Internal nodes: **operators**
  - Leaves: **operands**
  - The tree structure reflects the **order of operations**
- **Example:** arithmetic expression tree for `((2 × (5 - 1)) + (3 × 2))`

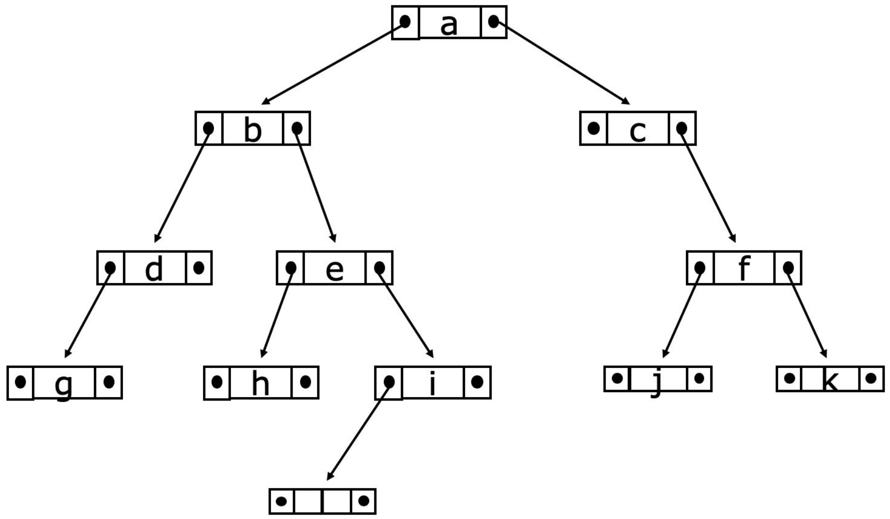


Tree Traversal

To **traverse** (or **walk**) the binary tree means to *visit each node in the binary tree exactly once*

Since a binary tree has three "parts", there are six possible ways to traverse the binary tree:

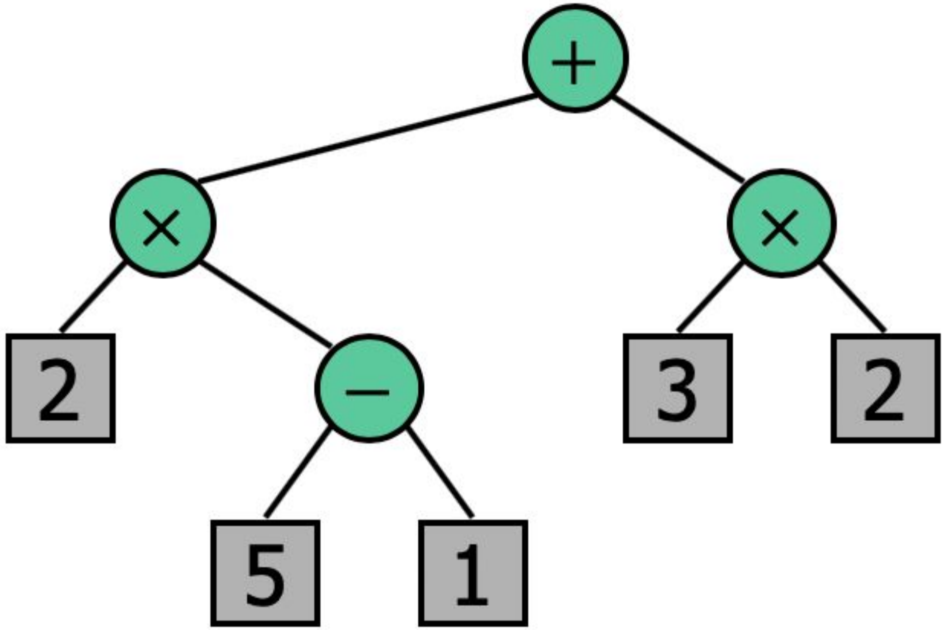
Preorder:	○ root, left, right	○ root, right, left
In-order:	○ left, root, right	○ right, root, left
Postorder:	○ left, right, root	○ right, left, root



Exercise: print the values of tree nodes in the sequence of each traversal

Arithmetic Expression Tree: Exercises

- Given an arithmetic expression tree, print the **fully parenthesized expression**
- Given an arithmetic expression tree, **evaluate the expression**



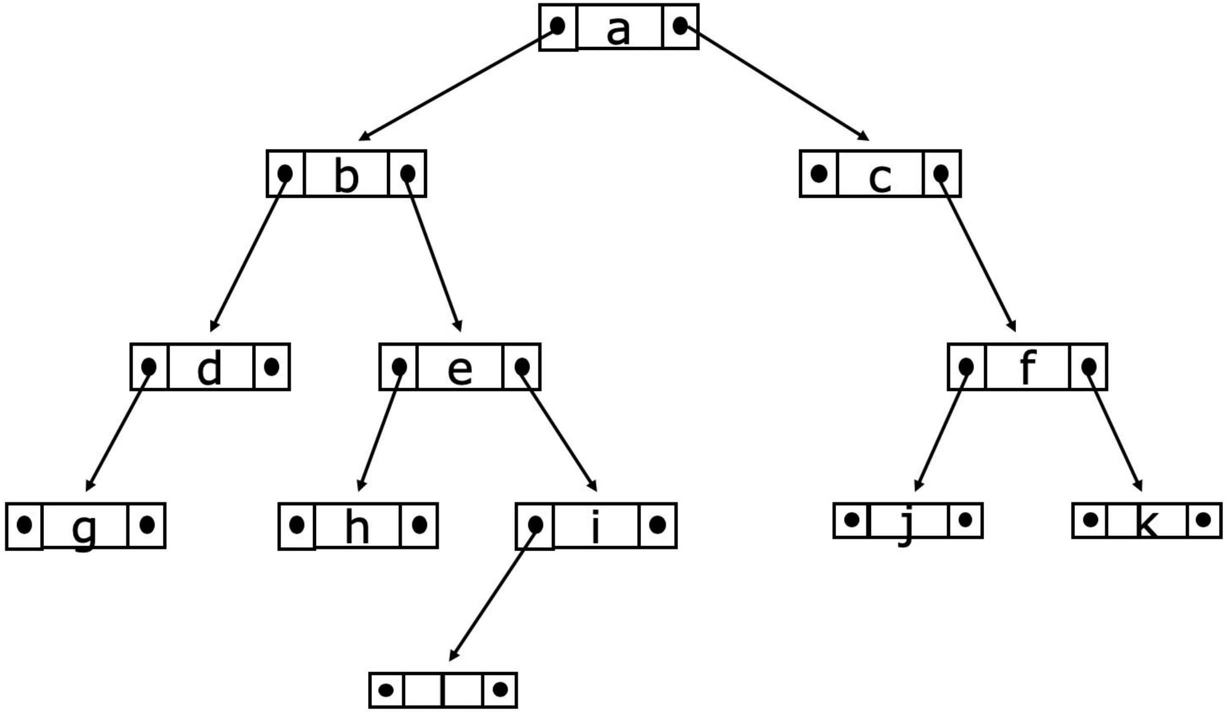
In-order -> Infix: `((2 × (5 - 1)) + (3 × 2))`

Post-order -> Postfix: `2 5 1 - * 3 2 * +`

Pre-order -> Prefix: `+ * 2 – 5 1 * 3 2`

Other Tree Exercises Traversals

- Given a binary tree, print the elements in **level order**
- How can you implement an **arbitrary tree** with binary trees?
- How can you you implement a binary tree with **arrays**?

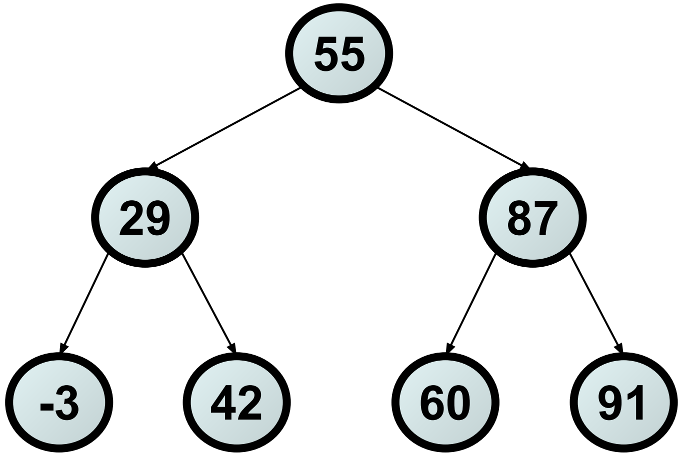


a b c d e f g h i j k l

Binary Search Trees

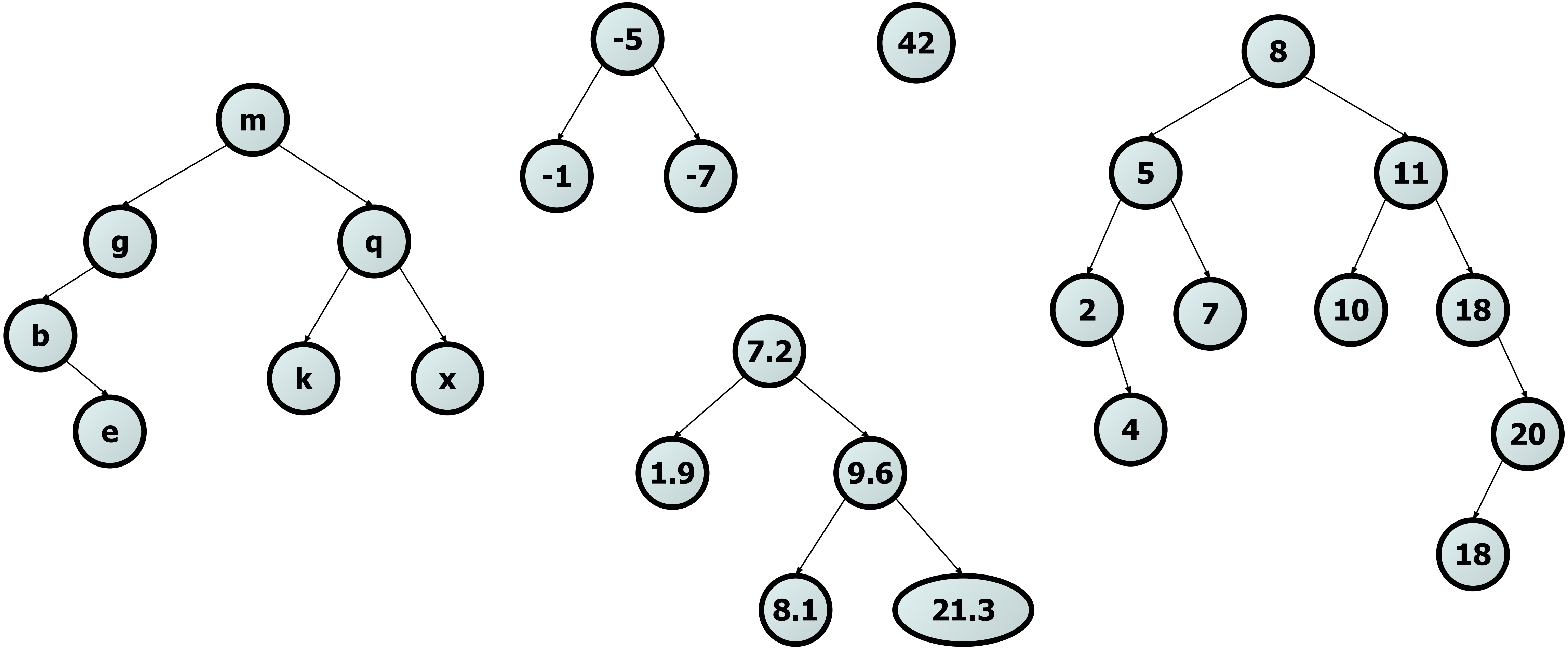
- **Binary Search Tree** ("BST"): a binary tree where **each node** R has the following properties:
  - o Every element of R's left subtree contains data **less than** R
  - o Every element of R's right subtree contains data **greater than** R

- BSTs store their elements in **sorted order**, which is helpful for searching/sorting tasks



BST Examples

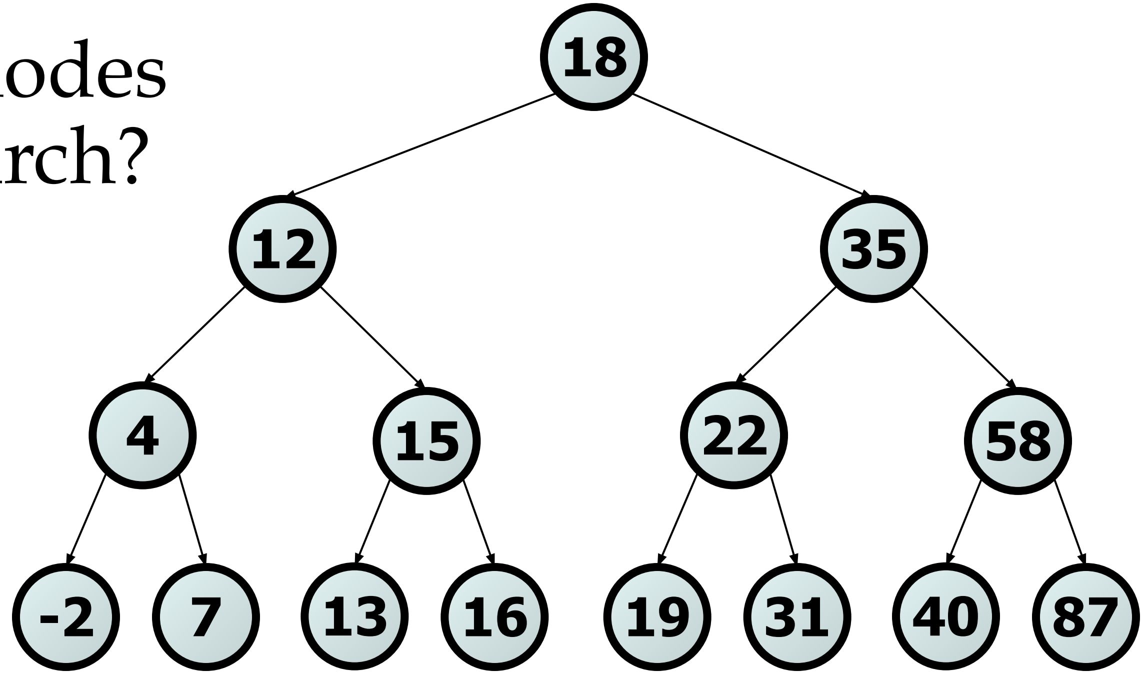
- Which of these trees are *binary search trees*?



Searching a BST

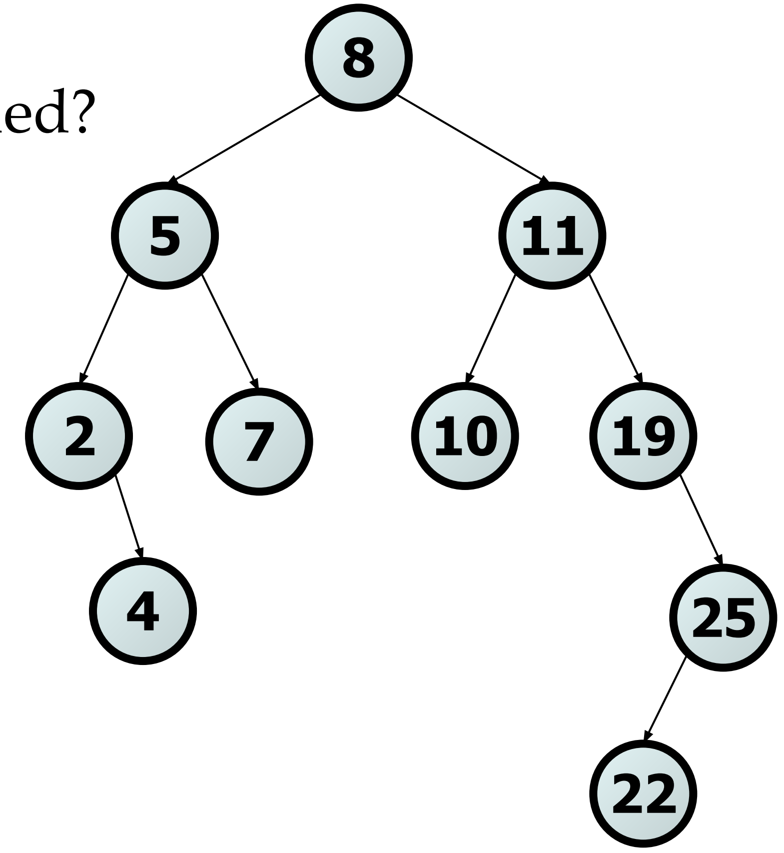
- Describe an algorithm for searching a binary search tree.
  - Try searching for the value 31
  - Then searching for value 6

- What is the maximum number of nodes you would visit to perform any search?



Adding to a BST

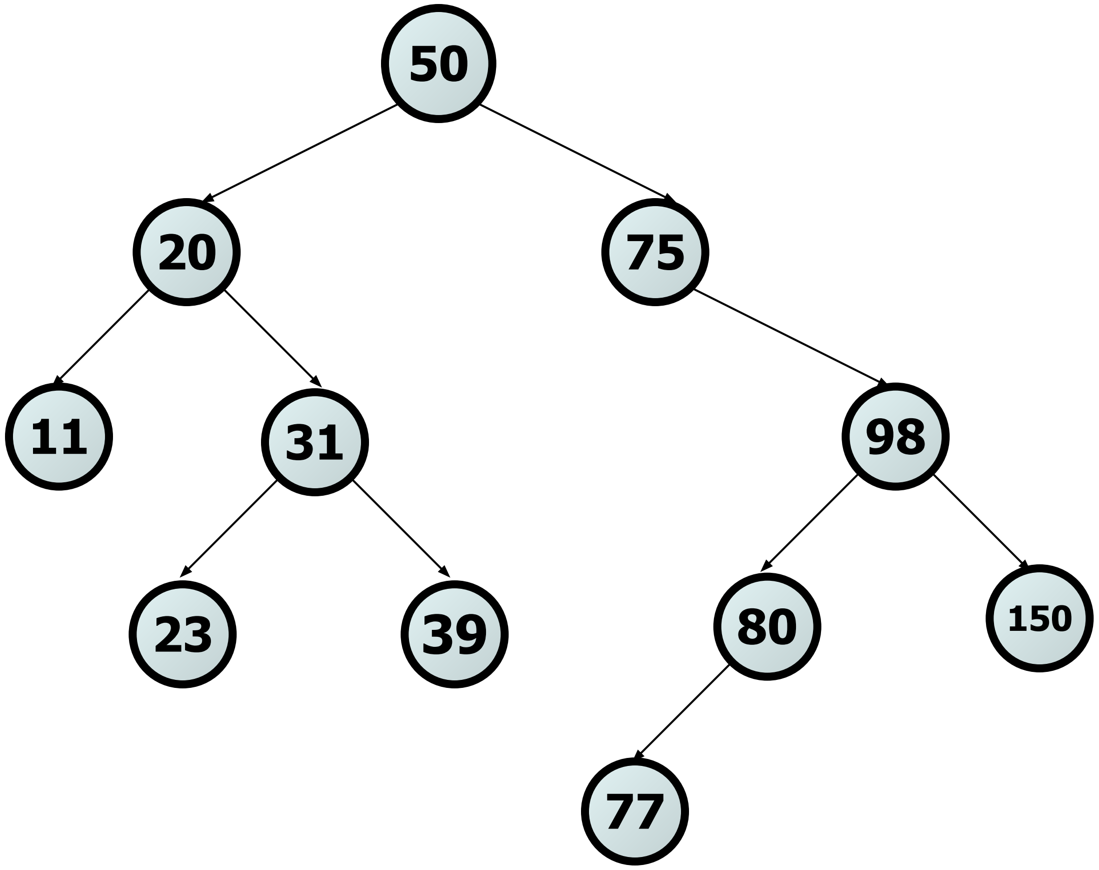
- Suppose we want to add new values to this BST
  - o Where should the value 14 be added?
  - o Where should 3 be added? 7?
  - o If the tree is empty, where should a new value be added?
- **What is the general algorithm?**
- **What is the time complexity?**



Adding Exercise

- Draw what a binary search tree would look like if the following values were added to an initially empty tree in this order:

50
20
75
98
80
31
150
39
23
11
77



Deleting from a BST

How can we delete an item from a BST?

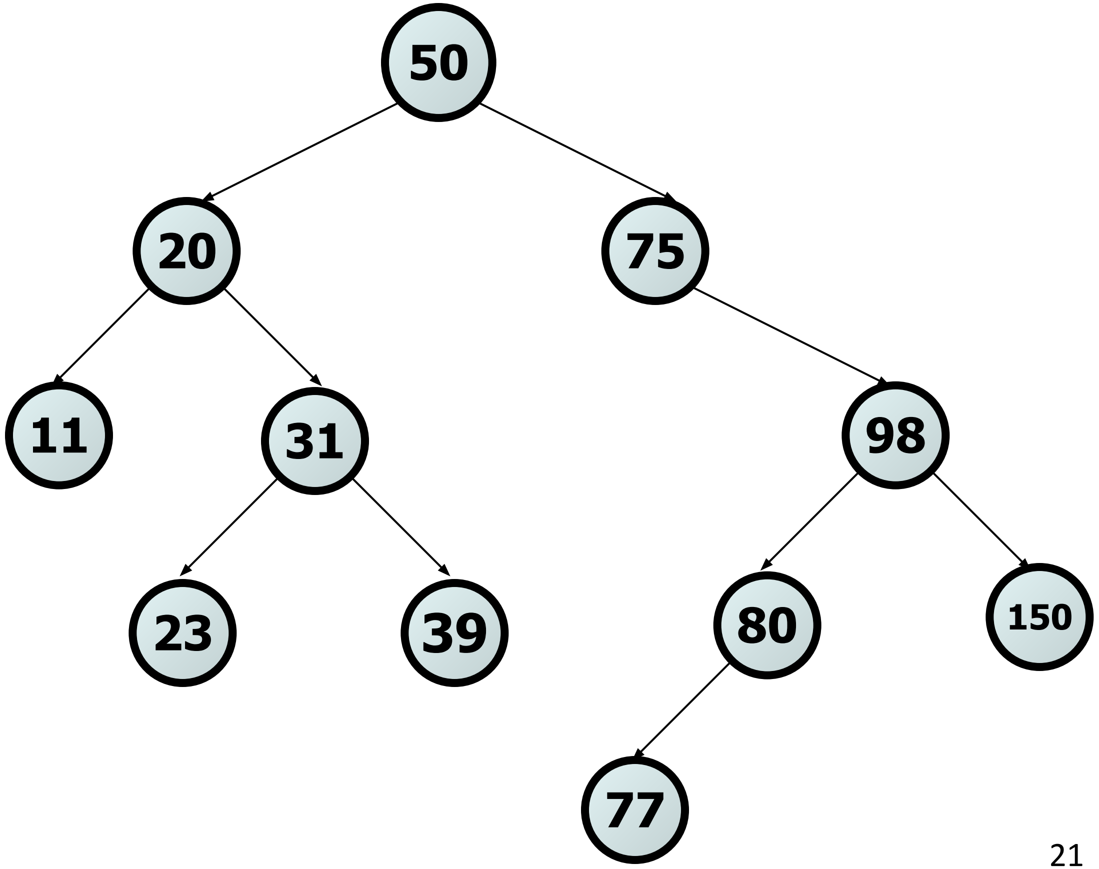
11

23

80

75

98



General algorithm?