CSC 1052 – Algorithms & Data Structures II: Trees

Professor Henry Carter
Spring 2017

CATS Reports

- Now available online!
  - Go to MyNova -> eLearn -> Course Evaluations
- Take 10 minutes
- Will not be visible to me until after grades are submitted
- Help me improve the course!

Recap

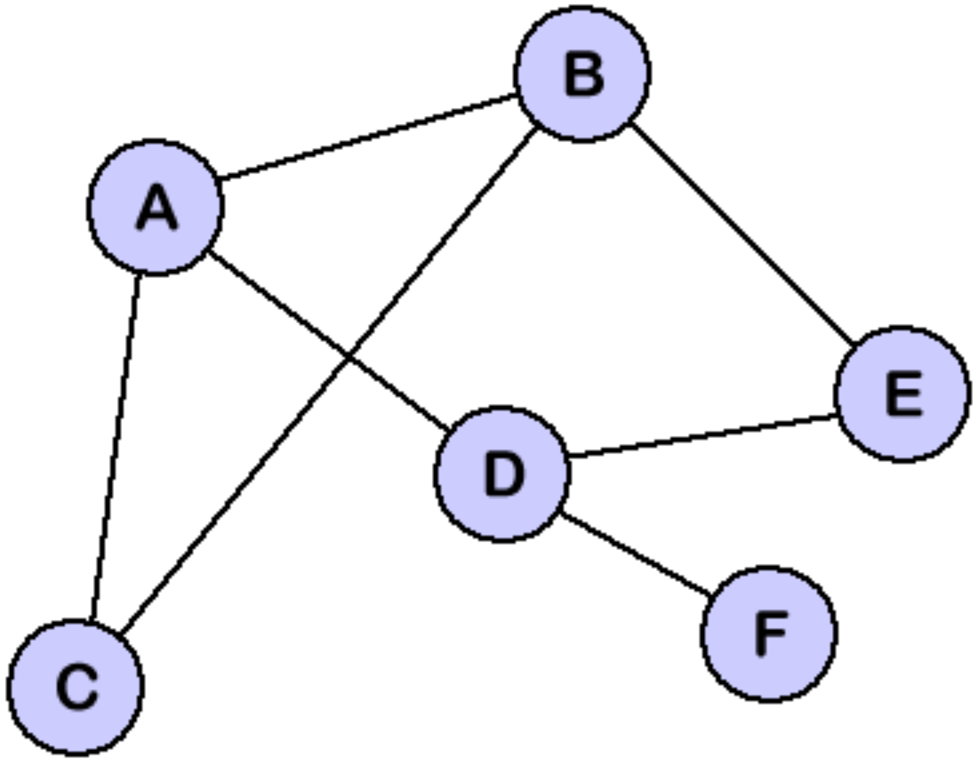
- Lists maintain a linear ordering of objects
- Lists can be used like previous data structures but more flexibly
- Iterator objects allow for simple iteration through list elements
- Lists can be used for a wide range of applications
    - Example: card deck
    - Consider: anywhere an array could be used without null gaps in entries

Binary Search Trees

- A new take on the collection ADT
- Recall: what were the advantages of the SortedArrayCollection?
- Recall: what were the advantages of the LinkedCollection
- Goal: Construct a linked data structure that allows for faster searching

Graphs

- Two sets: vertices (nodes) and edges (links)
- Trees: a subset of graphs that:
  - Have no cycles
  - Every node has one parent
  - Subtrees are disjoint
- Differs from previous constructions in that it *is nonlinear*

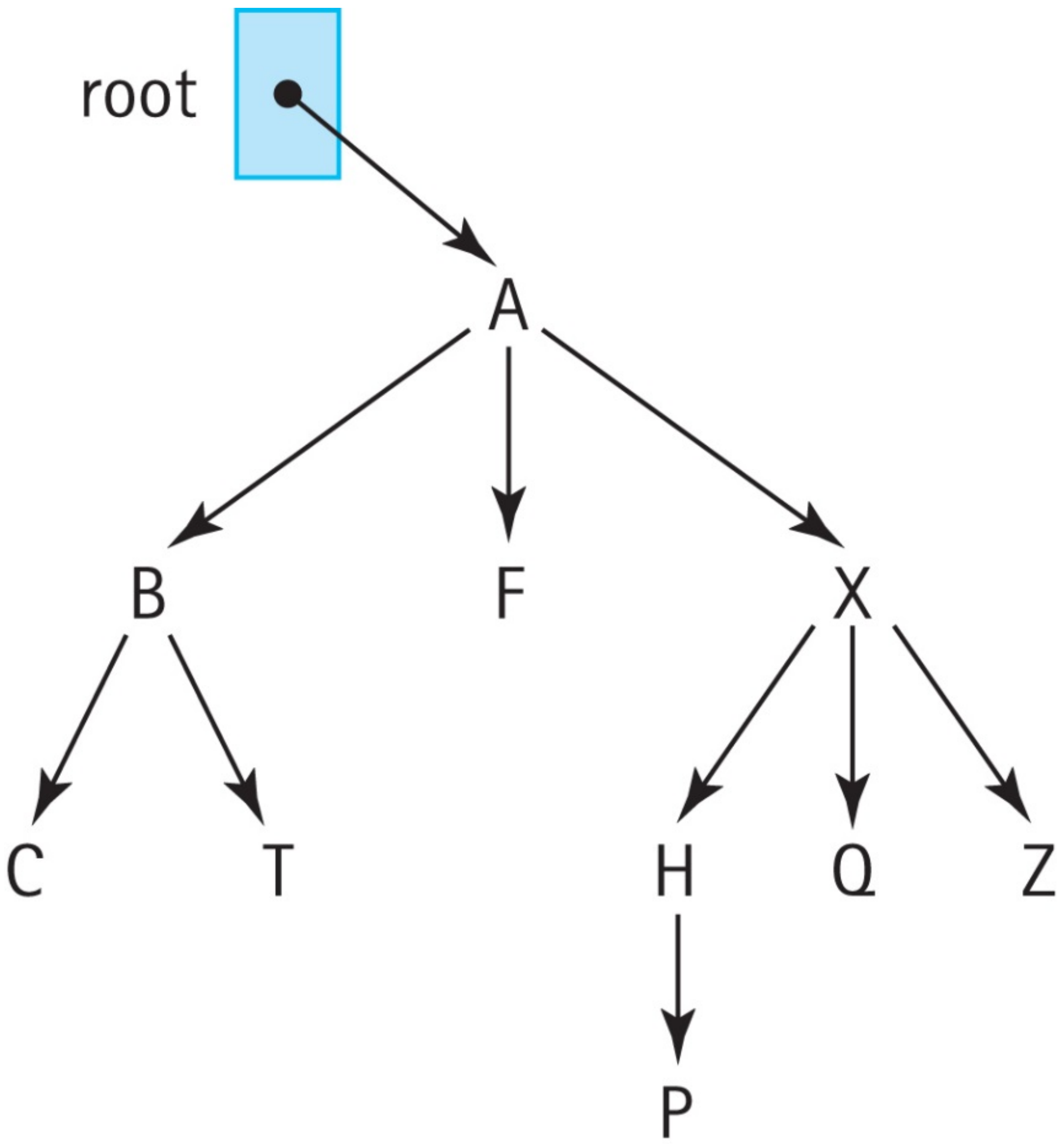


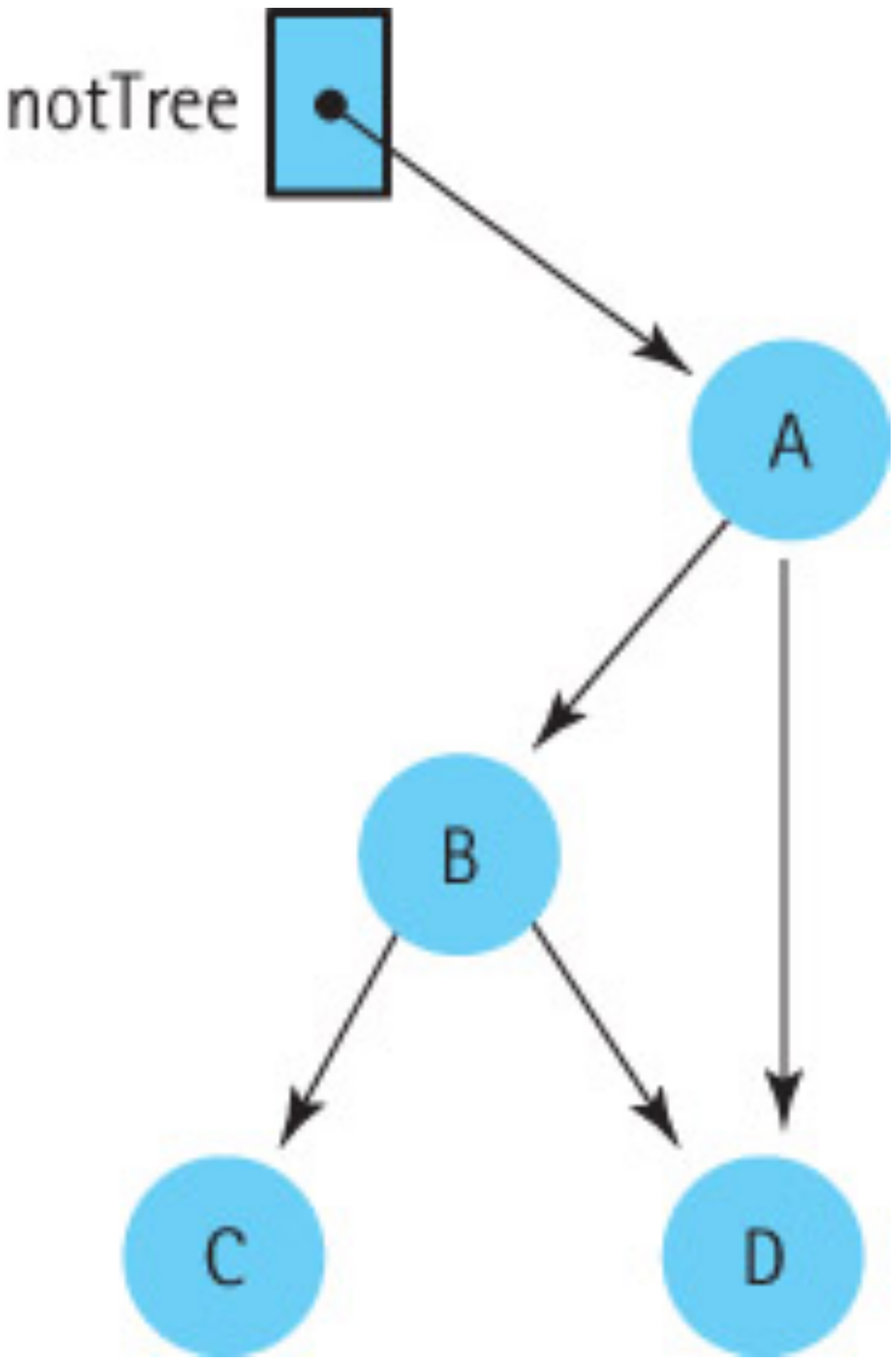
Terminology

- Root
- Parent
- Child
- Sibling
- Leaf
- Subtree
- Descendants
- Ancestors

"This is gobbledygook. I asked for mumbo-jumbo."

Example Tree

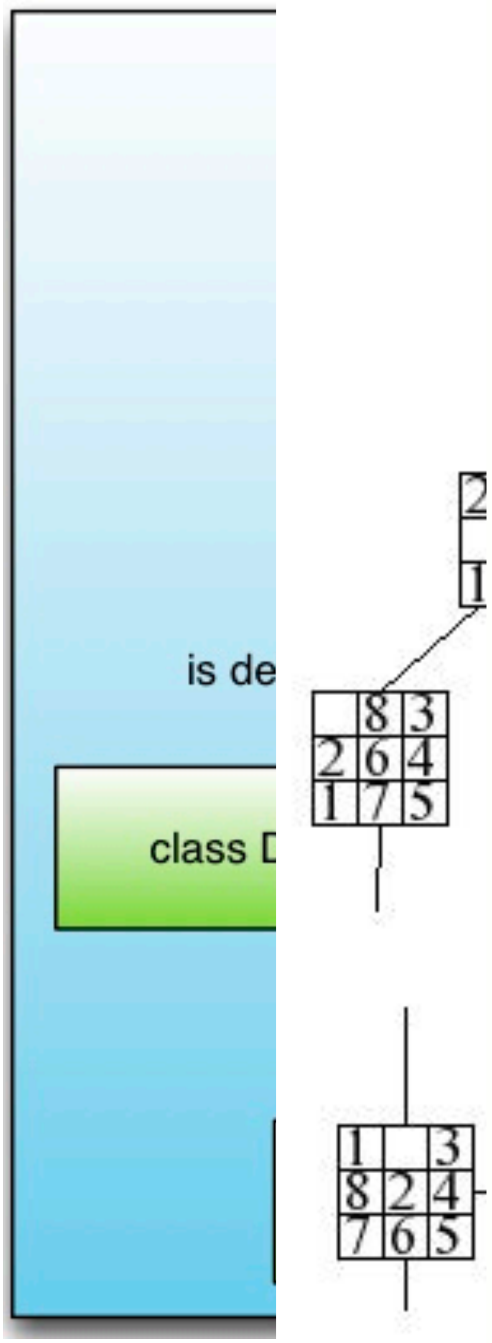


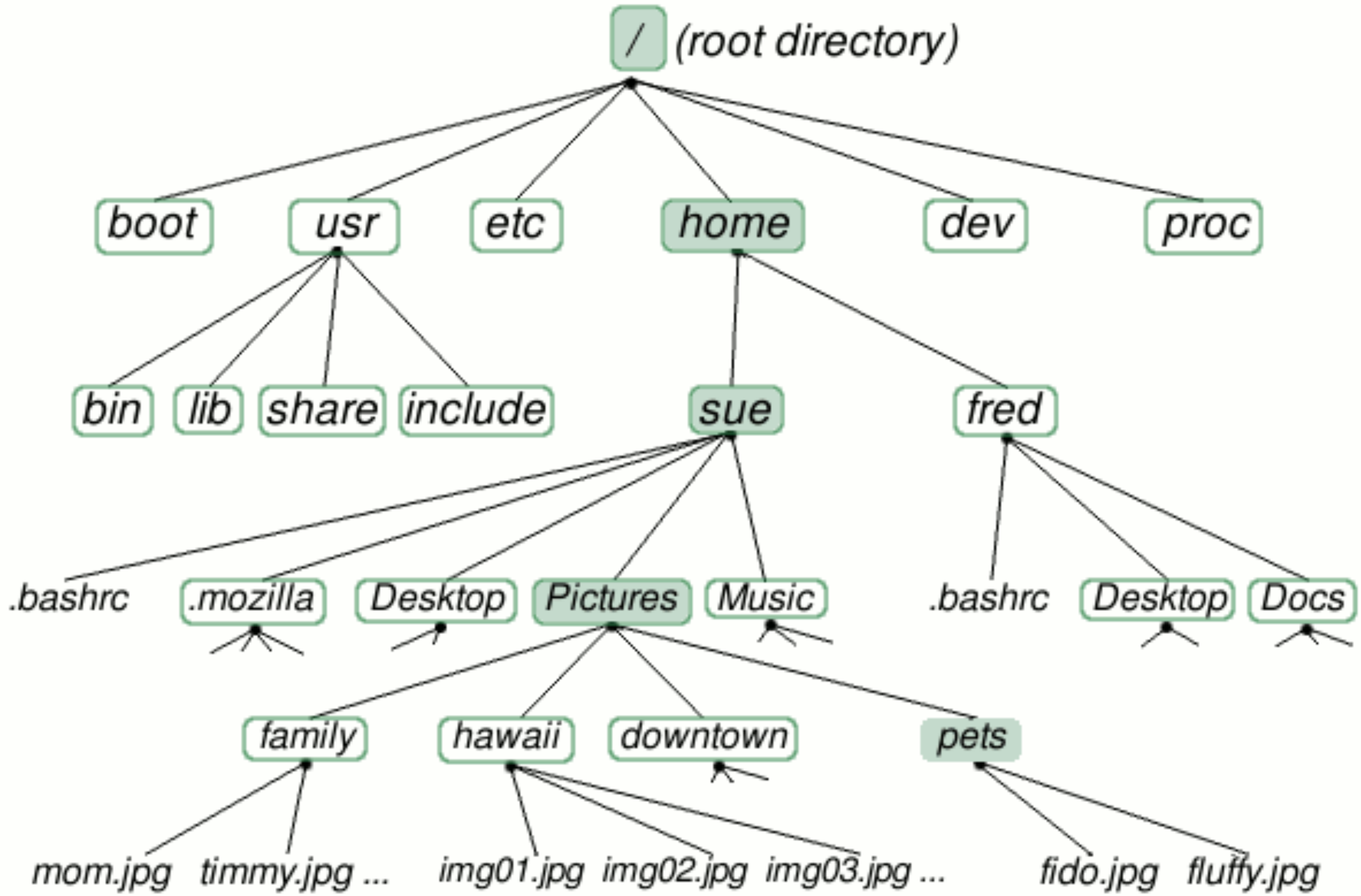


Why trees?

- Conveniently represents hierarchies
- Conveniently represents decision making
- Allows for faster search!

Example Trees

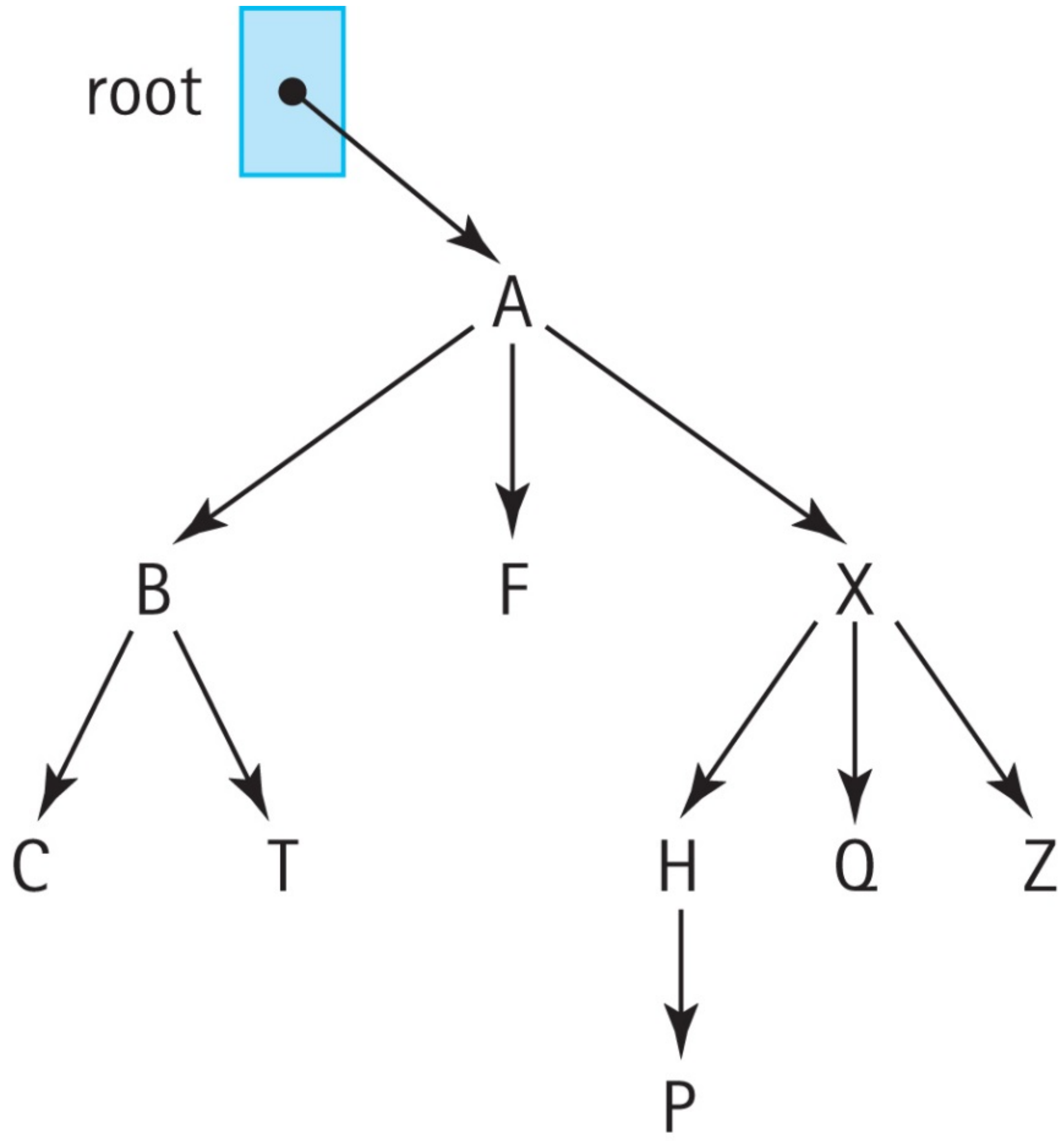




Traversals

- Given data stored in a tree, we often need to process the data as a whole
    - Print out the elements
    - Modify or sum values
    - Locate particular values
- For any graph, there are two traversal methods:
    - Breadth first
    - Depth first

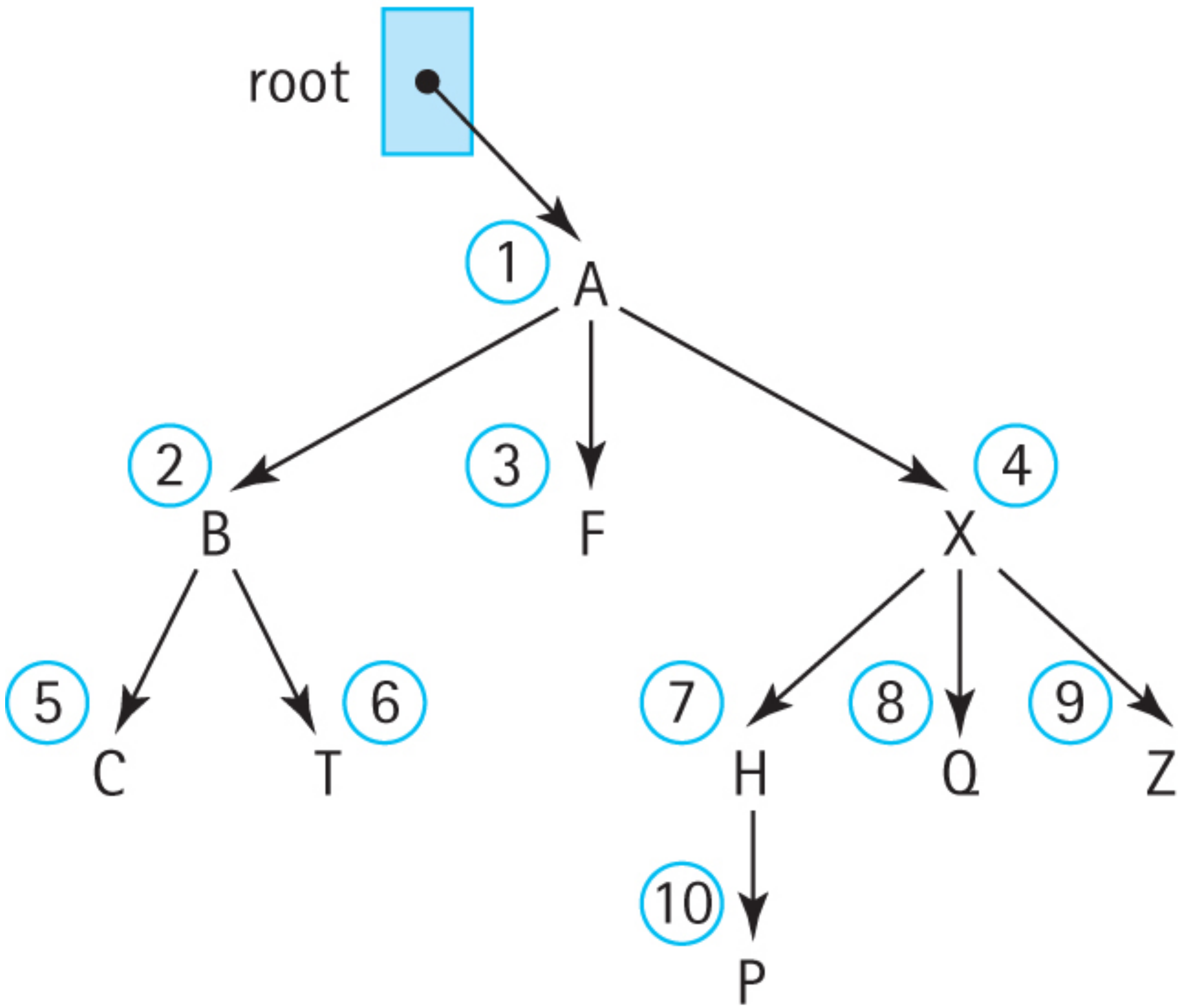
Breadth vs Depth



Breadth-First

- Process the root node
- Process all of the root's children
- In order, process each of the root's grandchildren
- After processing the root, add the children to an ordered data structure and process in that order
    - What ordered ADT?

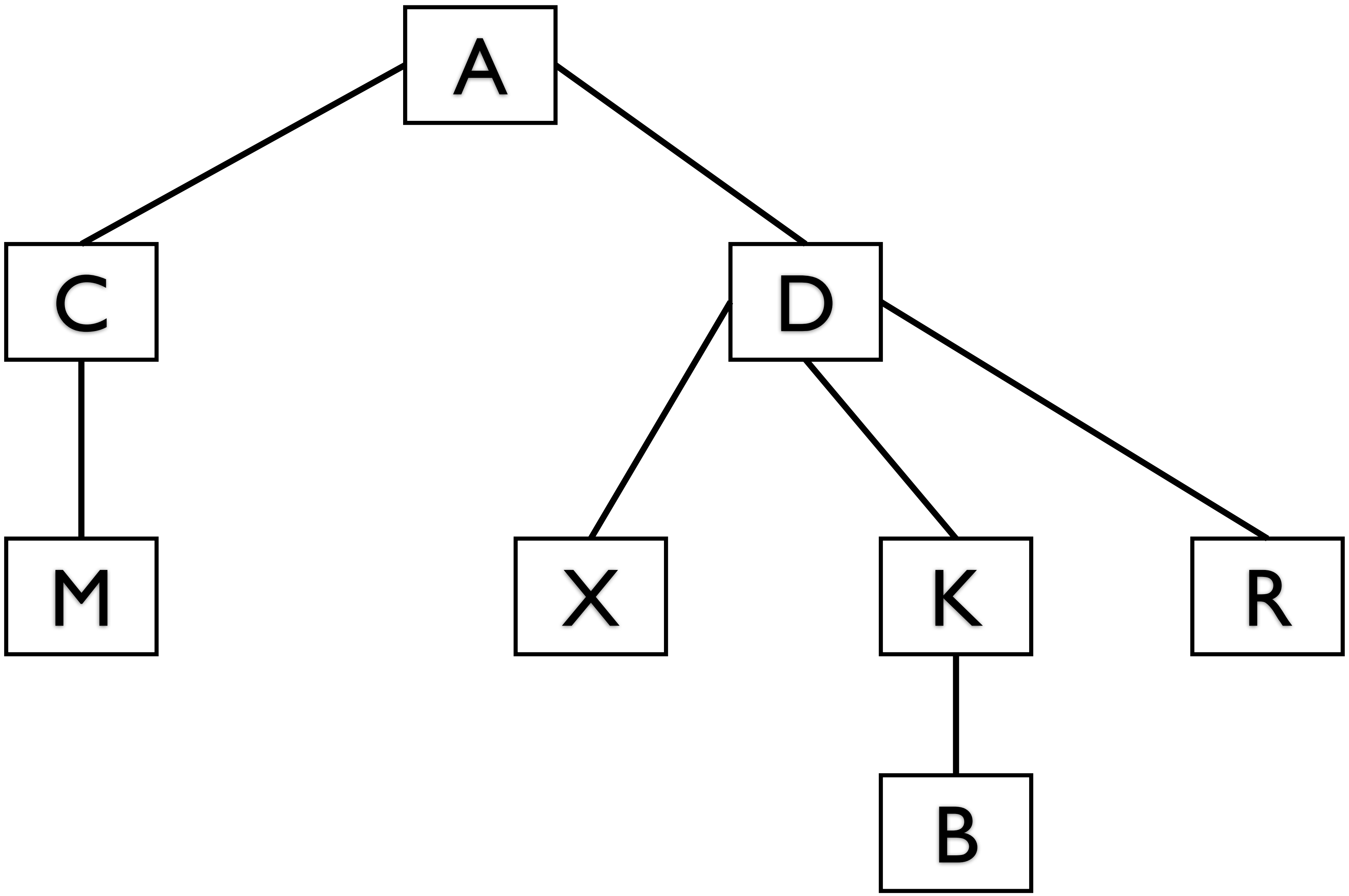
BFS Illustration



BFS Code

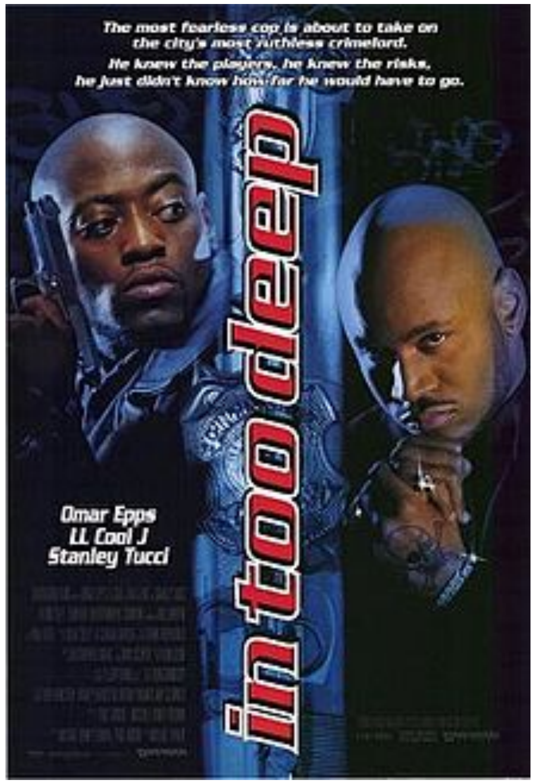
```
Breadth-First Traversal(root)

Instantiate a queue of nodes
if (root is not null)
{
   queue.enqueue(root)
   while (!queue.isEmpty())
   {
      node = queue.dequeue()
      Visit node
      Enqueue the children of node
         (from left to right) into queue
   }
}
```

Practice!

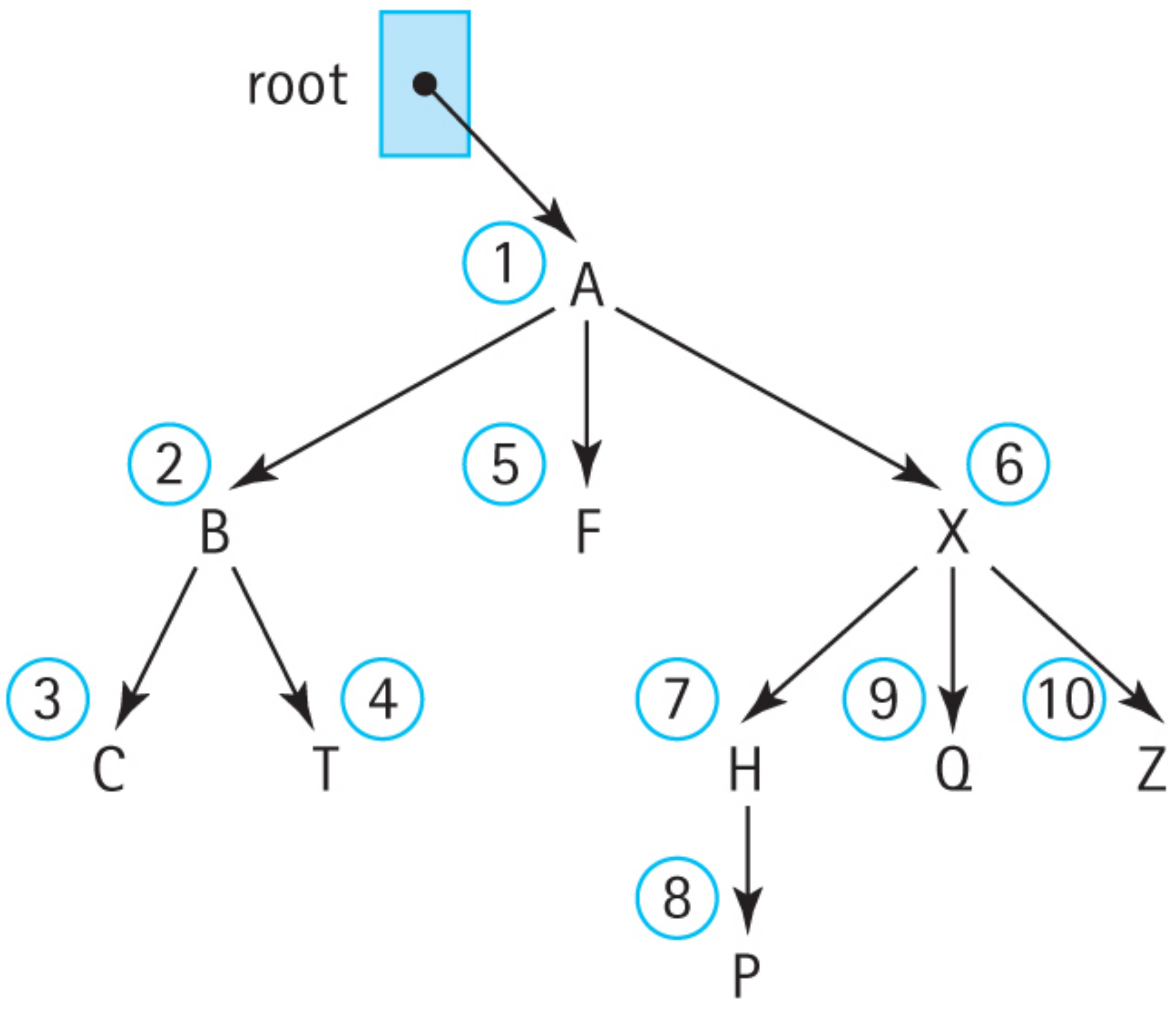


Depth-First

- Process to the farthest node from the root on the *left*
- When we hit a leaf, backtrack
- Resume moving *away* as soon as a new path is found
- As we descend into the tree, store the nodes going down and backtrack in *reverse order*
  - What ordered ADT?



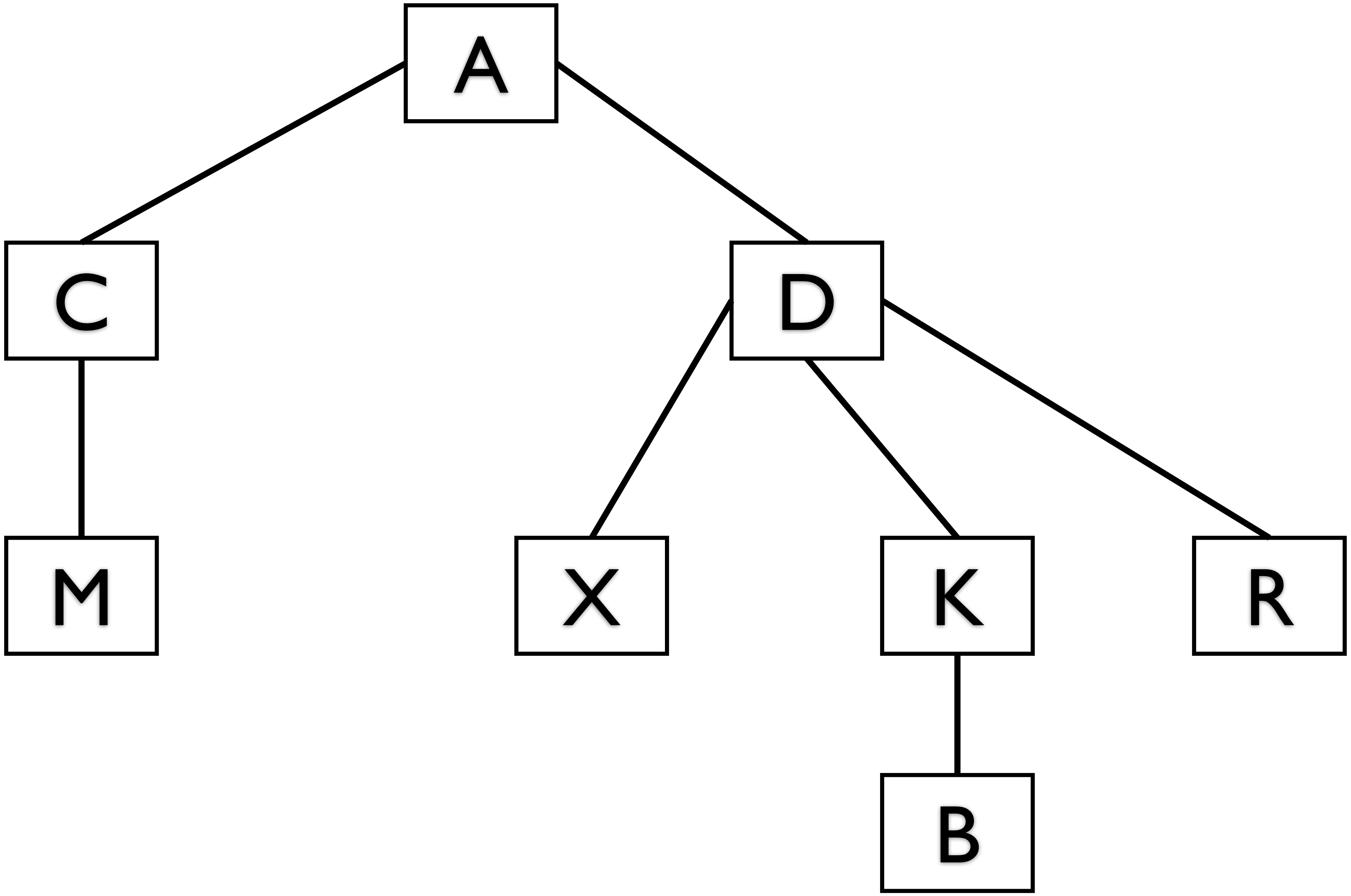
DFS Illustration



DFS Code

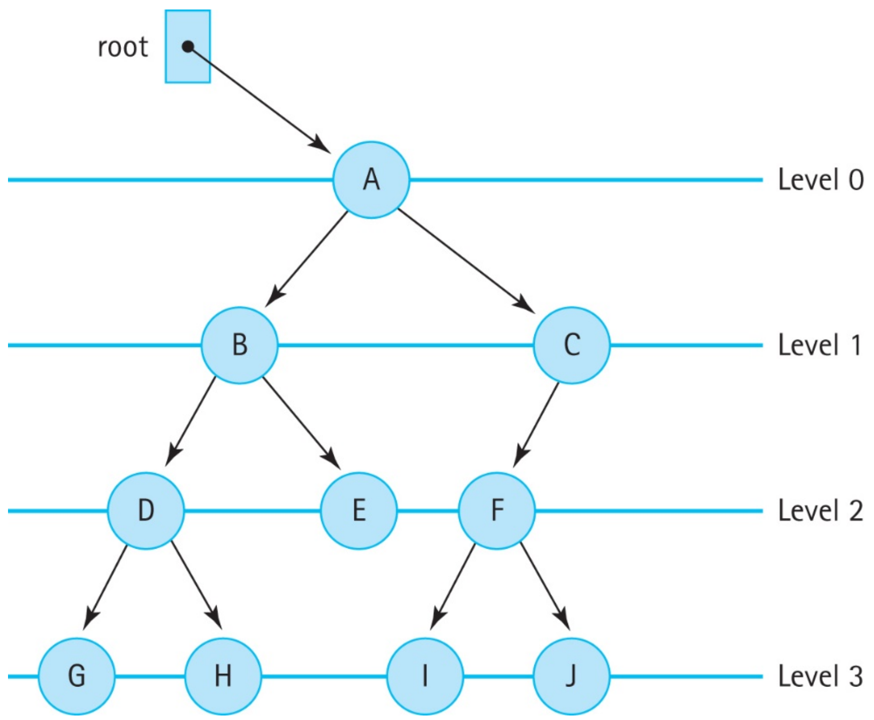
```
Depth-First Traversal(root)

Instantiate a stack of nodes
if (root is not null)
{
   stack.push(root)
   while (!stack.isEmpty())
   {
      node = stack.top()
      stack.pop()
      Visit node
      Push the children of node
         (from right to left) onto queue
   }
}
```

Practice!

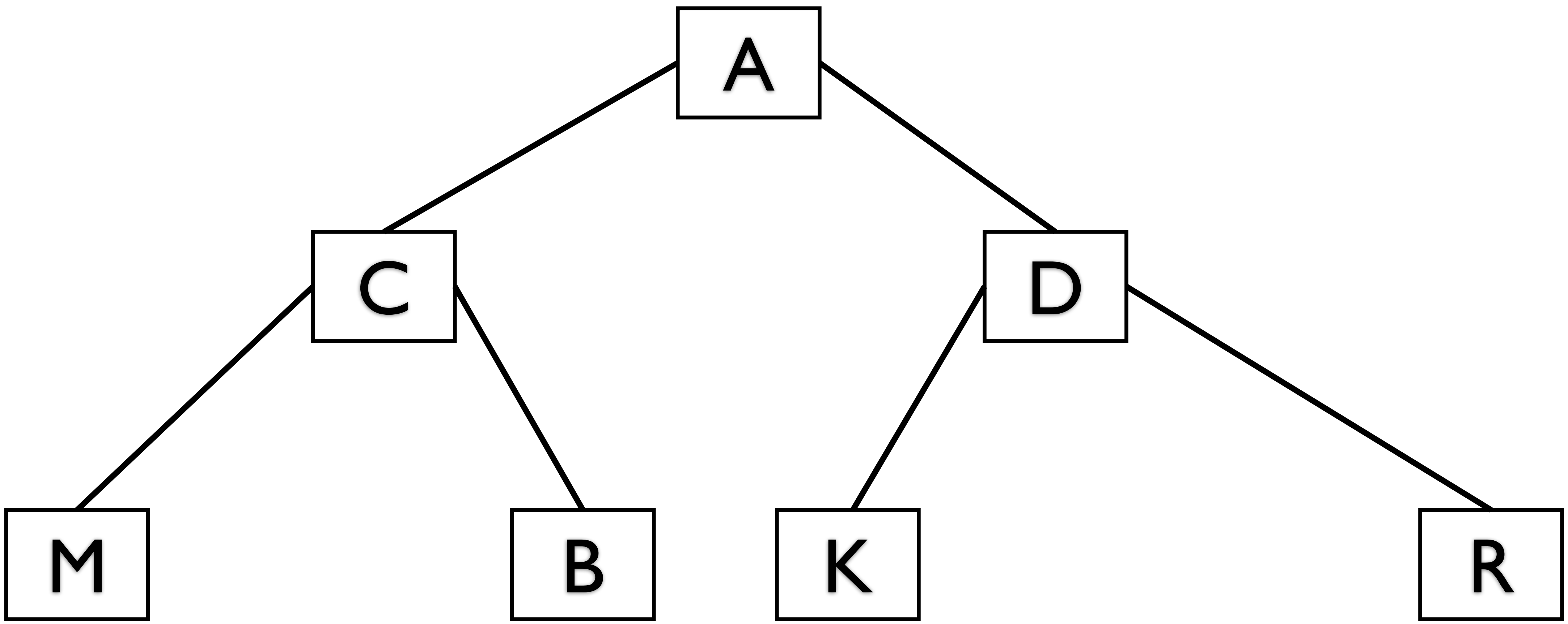


Binary Trees

- A tree with an additional restriction:
  - Every node may have *at most* two children
- Allows the number of nodes at each level to double
- Doubling number of nodes (may) keep the tree shallow



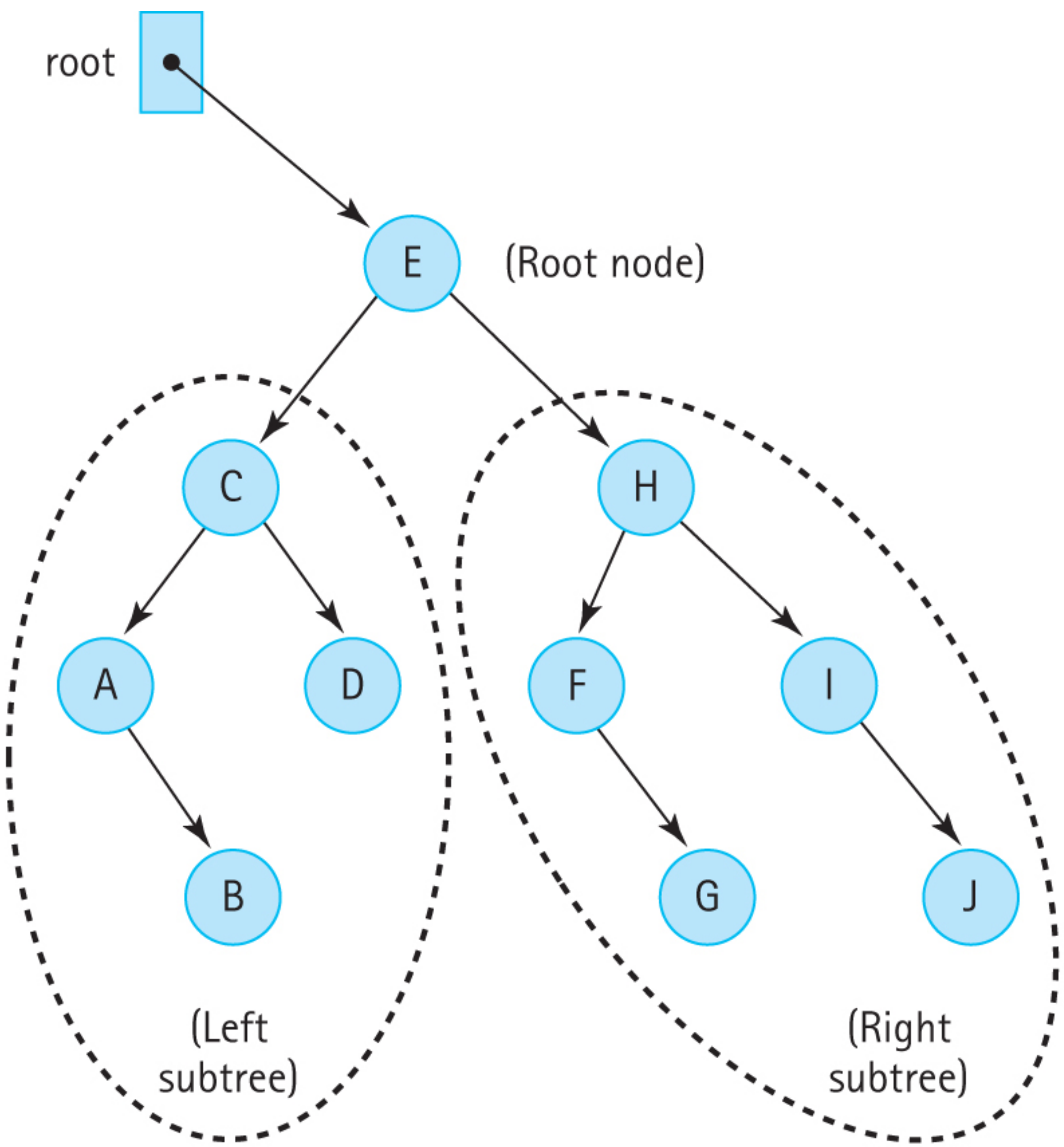
A Binary Tree



Binary Search Trees

- Add another condition:
  - All nodes in the left subtree are less than the root
  - All nodes in the right subtree are greater than the root
- Allows Searching within the tree elements
- Assuming the tree is balanced and as shallow as possible, how long until you find the element?

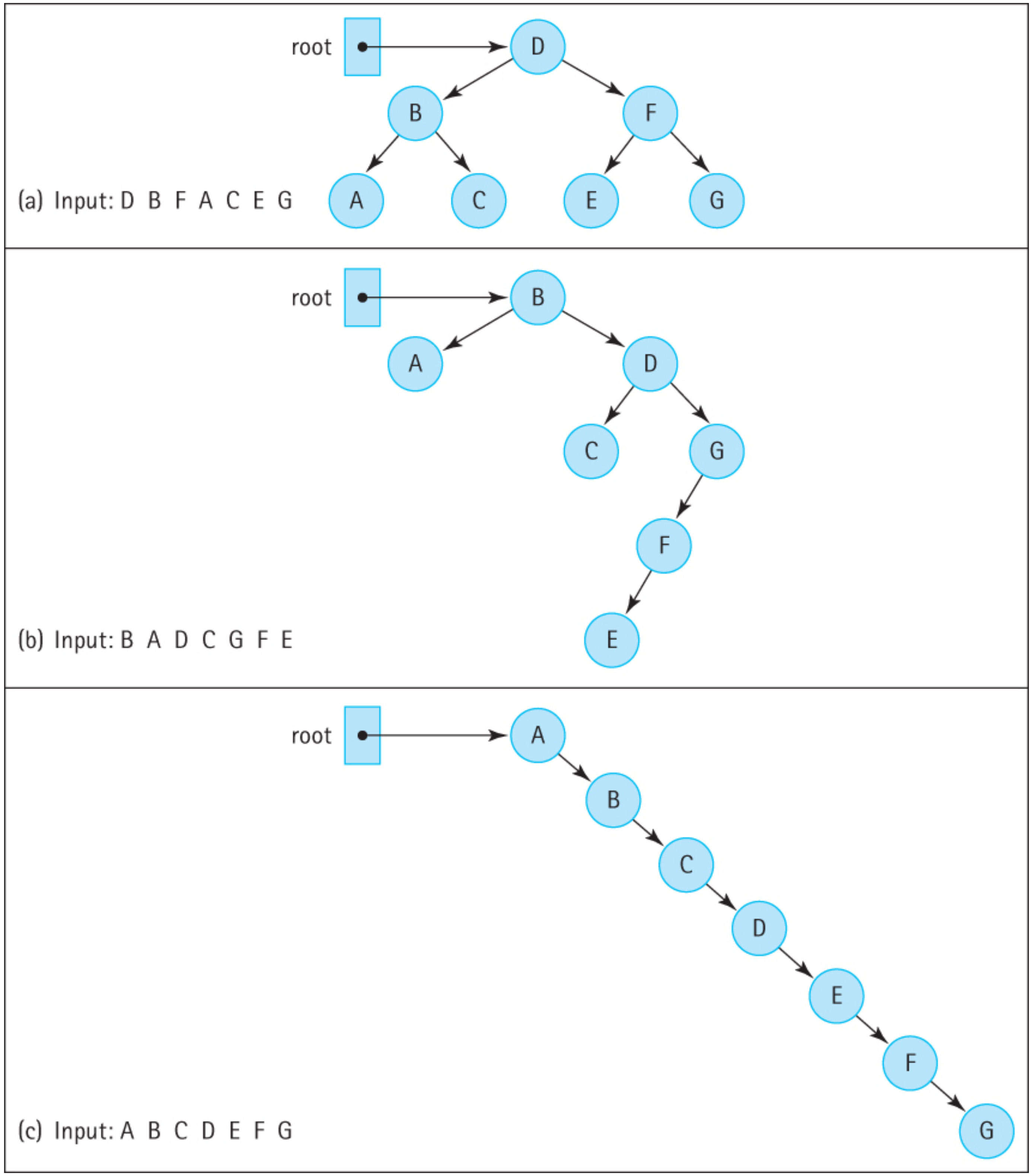
Example BST



All values in the left subtree are less than or equal to the value in the root node.

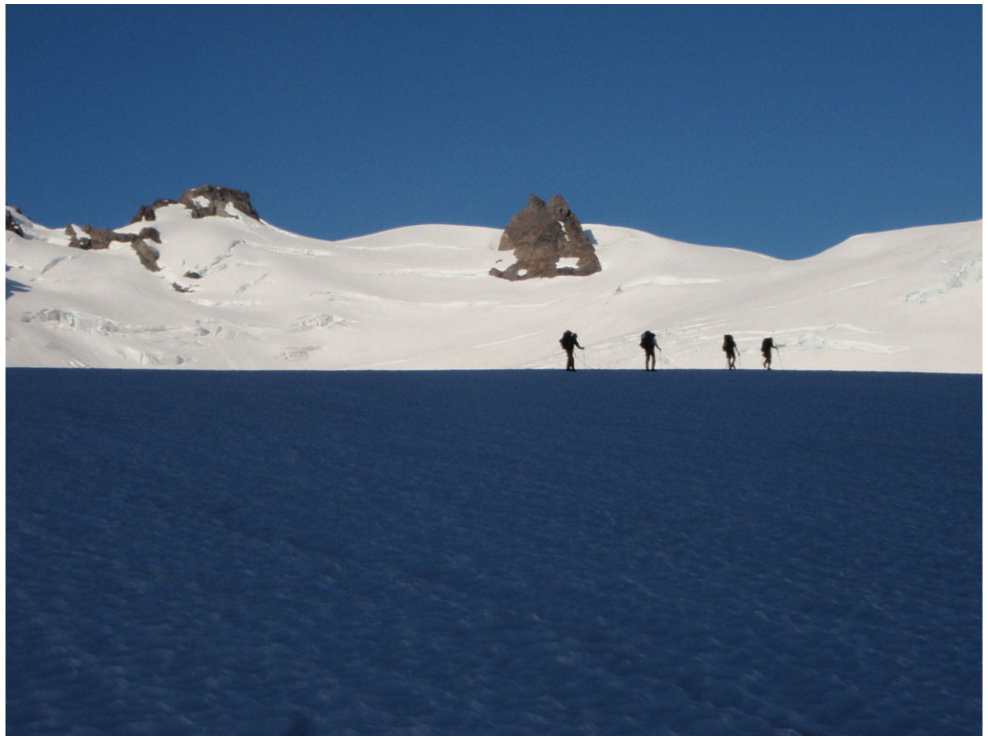
All values in the right subtree are greater than the value in the root node.

Balanced vs Unbalanced



Traversal Take 2

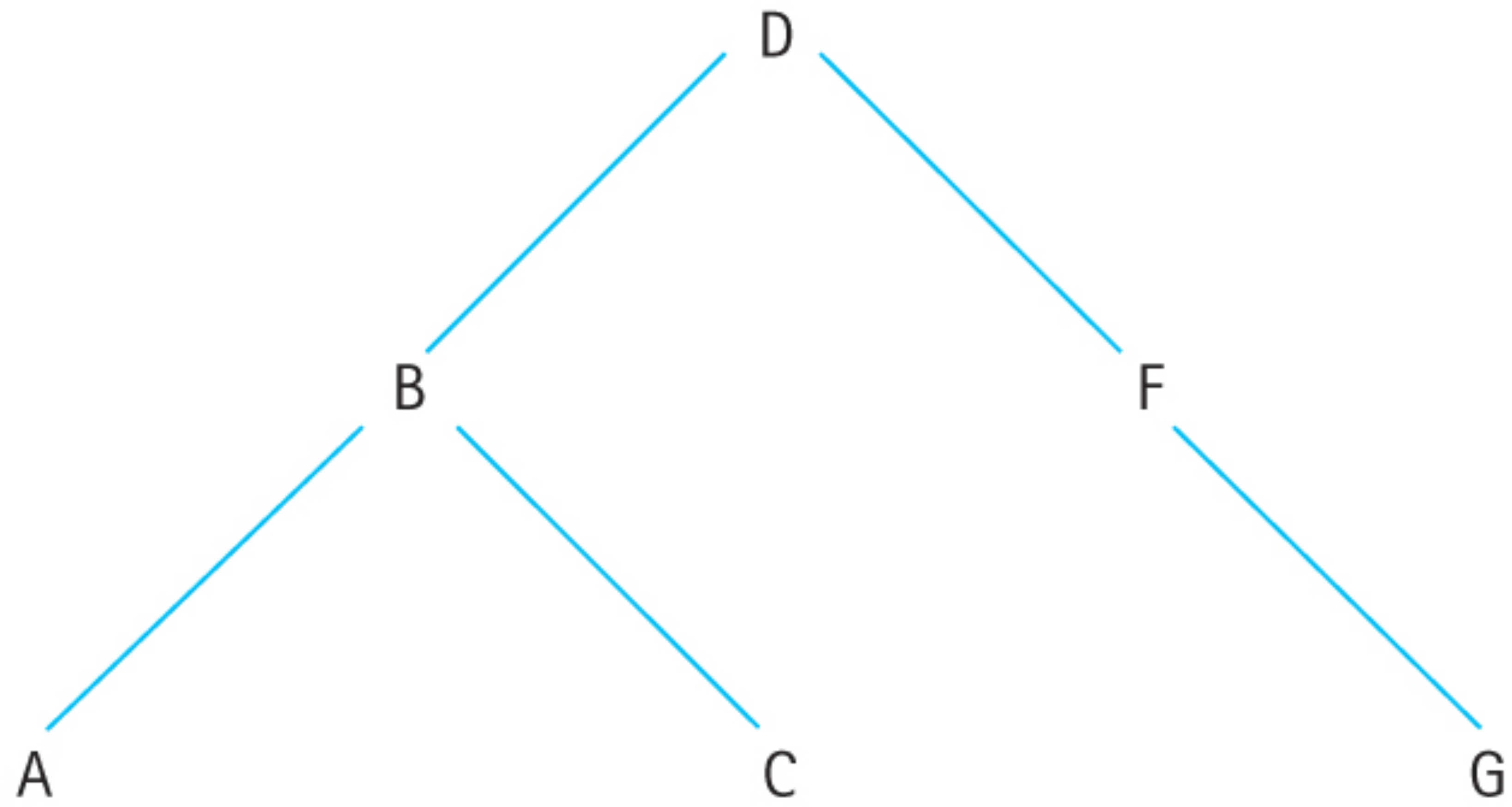
- Binary search trees allow three new traversal patterns
  - Preorder
  - Inorder
  - Postorder
- The order is determined by the ordering of processing left subtree, root, and right subtree



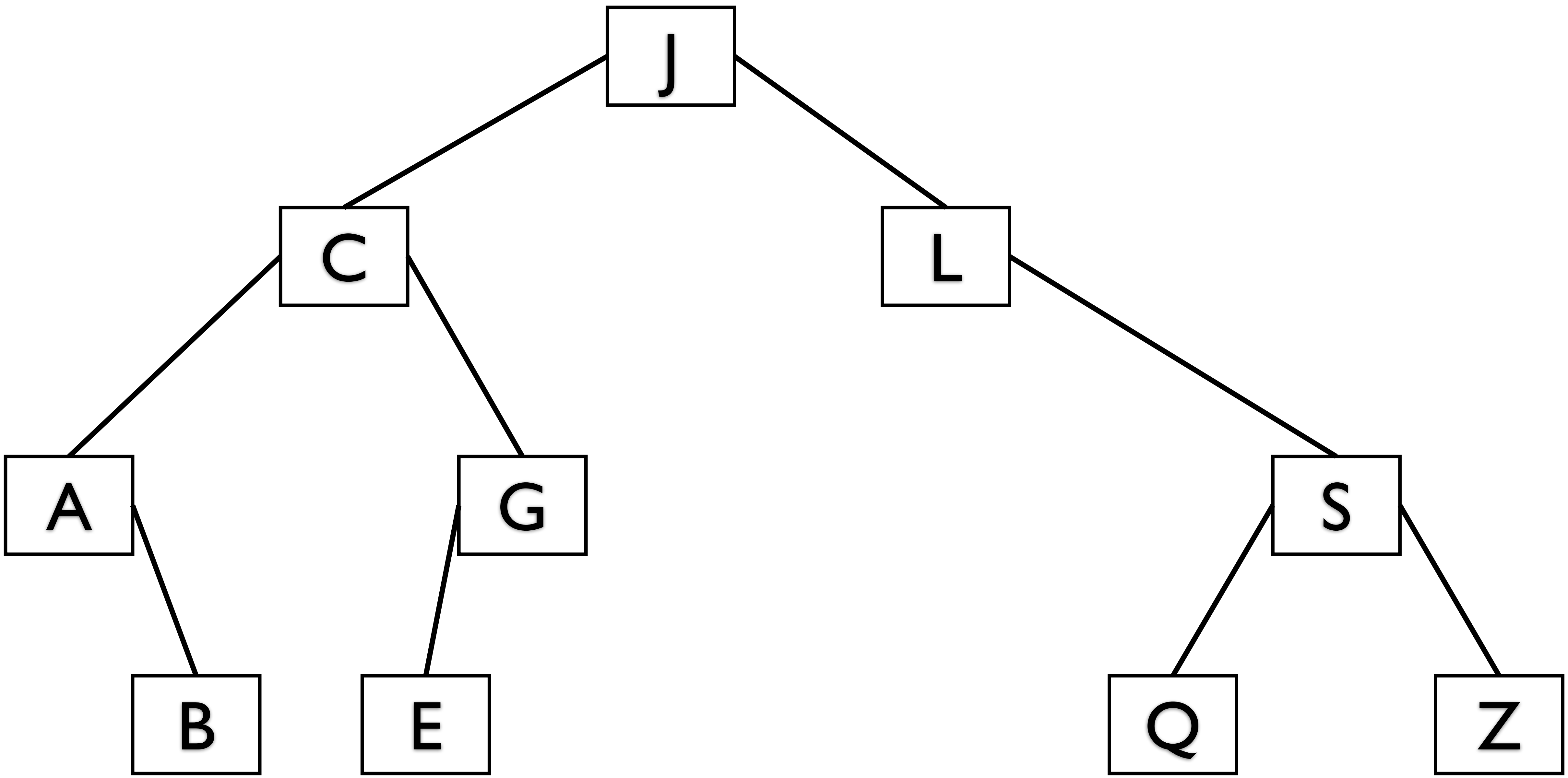
Ordering

- Preorder: visit the root first
  - Then left, right subtrees
- Inorder: visit the root second
  - Left subtree first, right third
- Postorder: visit the root last
  - Left and right subtrees first

Traversals



Practice



Recap

- Graphs represent structured data in a nonlinear fashion
  - Composed of nodes and links between the nodes
- Trees allow for storage of hierarchical data
  - Traversal in breadth-first or depth-first
- Binary Search Trees allow for a binary search to be embedded in the tree structure
  - Traversal in preorder, inorder, or postorder

Next Time...

- Dale, Joyce, Weems Chapter 7.3-5
  - Remember, you need to read it BEFORE you come to class!
- Check the course webpage for practice problems
- Peer Tutors
  - http://www.csc.villanova.edu/help/