Trees

Tree

Definition

- A *tree* is a finite nonempty set of elements
- It is an abstract model of a hierarchical structure.
- consists of nodes with a parent-child relation.

Applications

- Organization charts
- File systems
- Programming environments

Tree

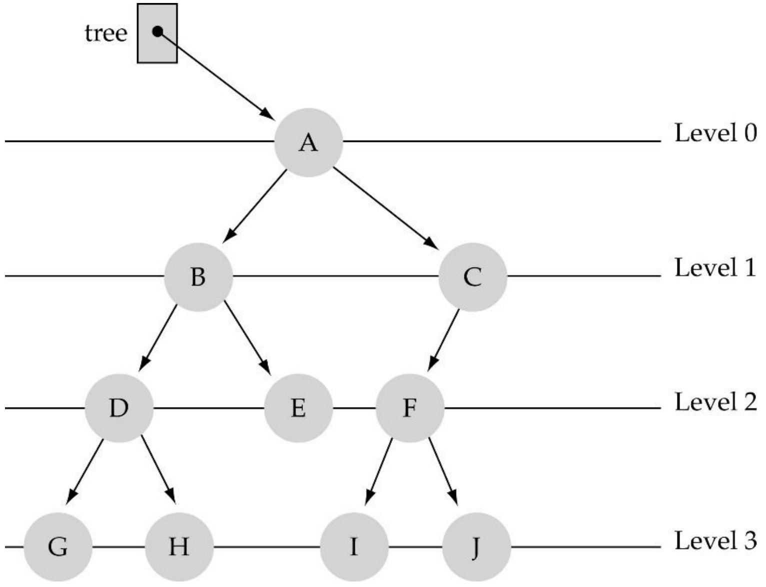
Terminology

- Root: node without parent (A)
- Siblings: nodes share the same parent
- Internal node: node with at least one child (A, B, C, F)
- External node (leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Descendant of a node: child, grandchild, grand-grandchild, etc.

Tree

Terminology

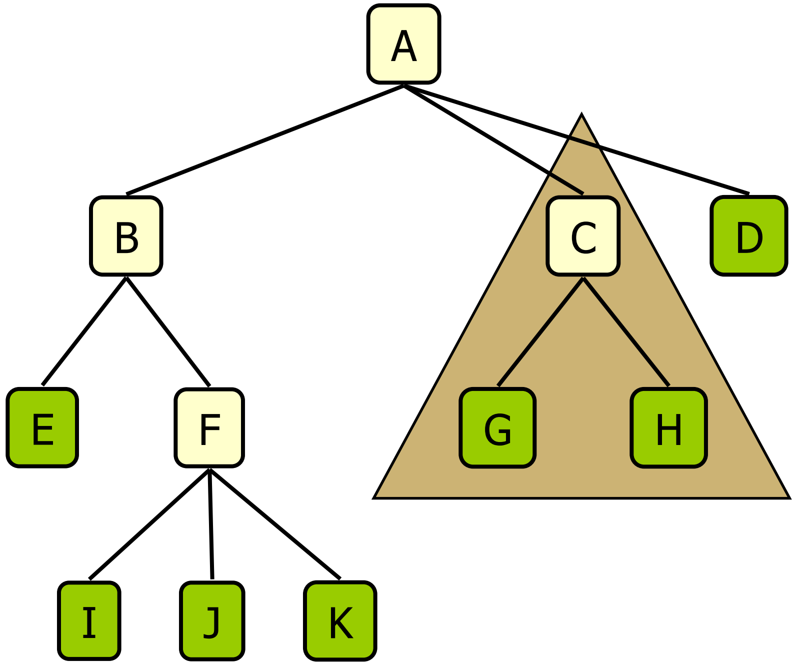
- Depth of a node:
  - number of ancestors *or*
  - its distance from the root
- Height of a tree: maximum depth of any node (3)
- Degree of a node: the number of its children
- Degree of a tree: the maximum number of its node.
- Subtree: tree consisting of a node and its descendants



Tree

Terminology

- Subtree: tree consisting of a node and its descendants

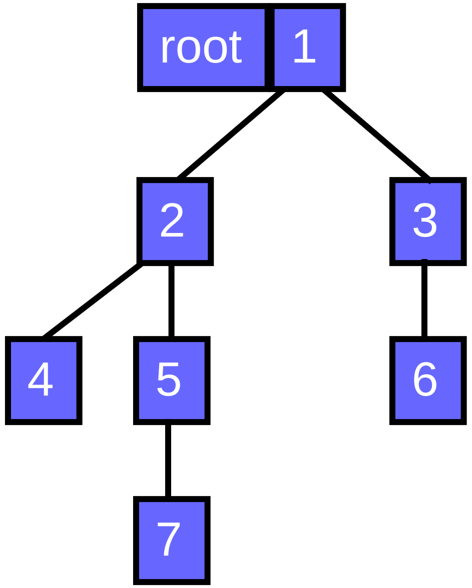


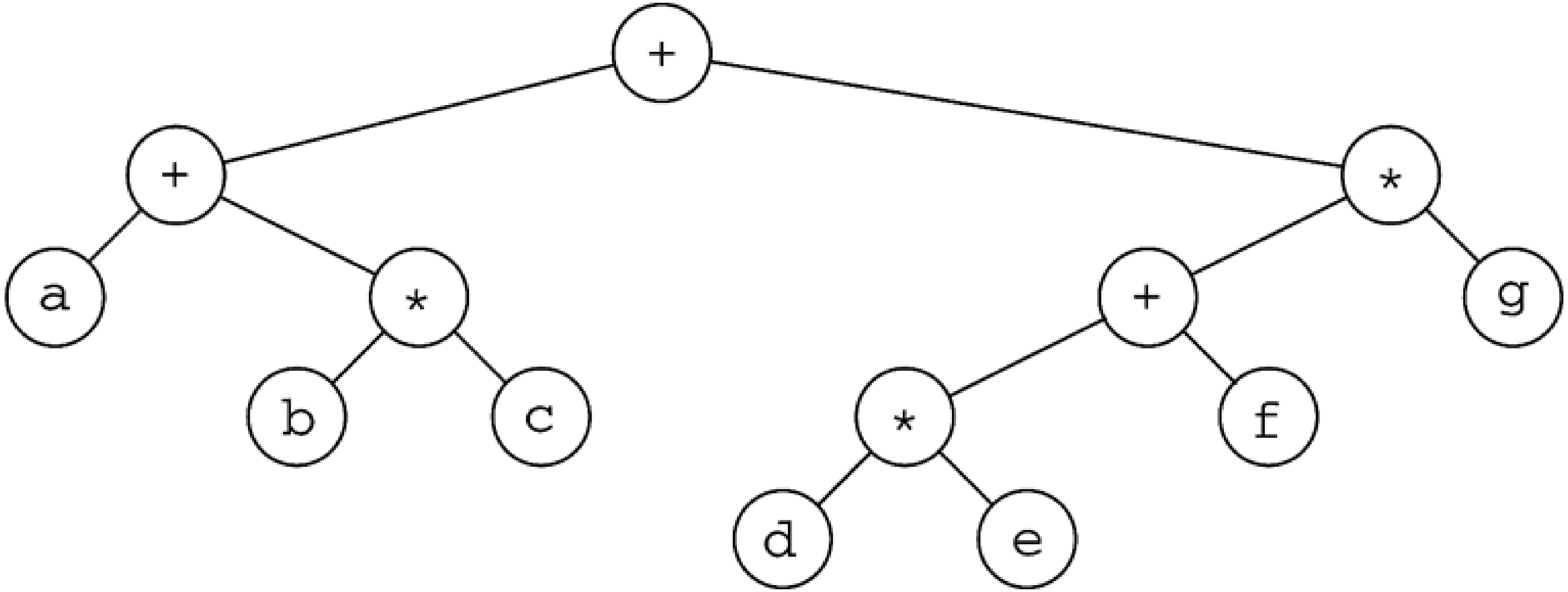
subtree

Binary Trees

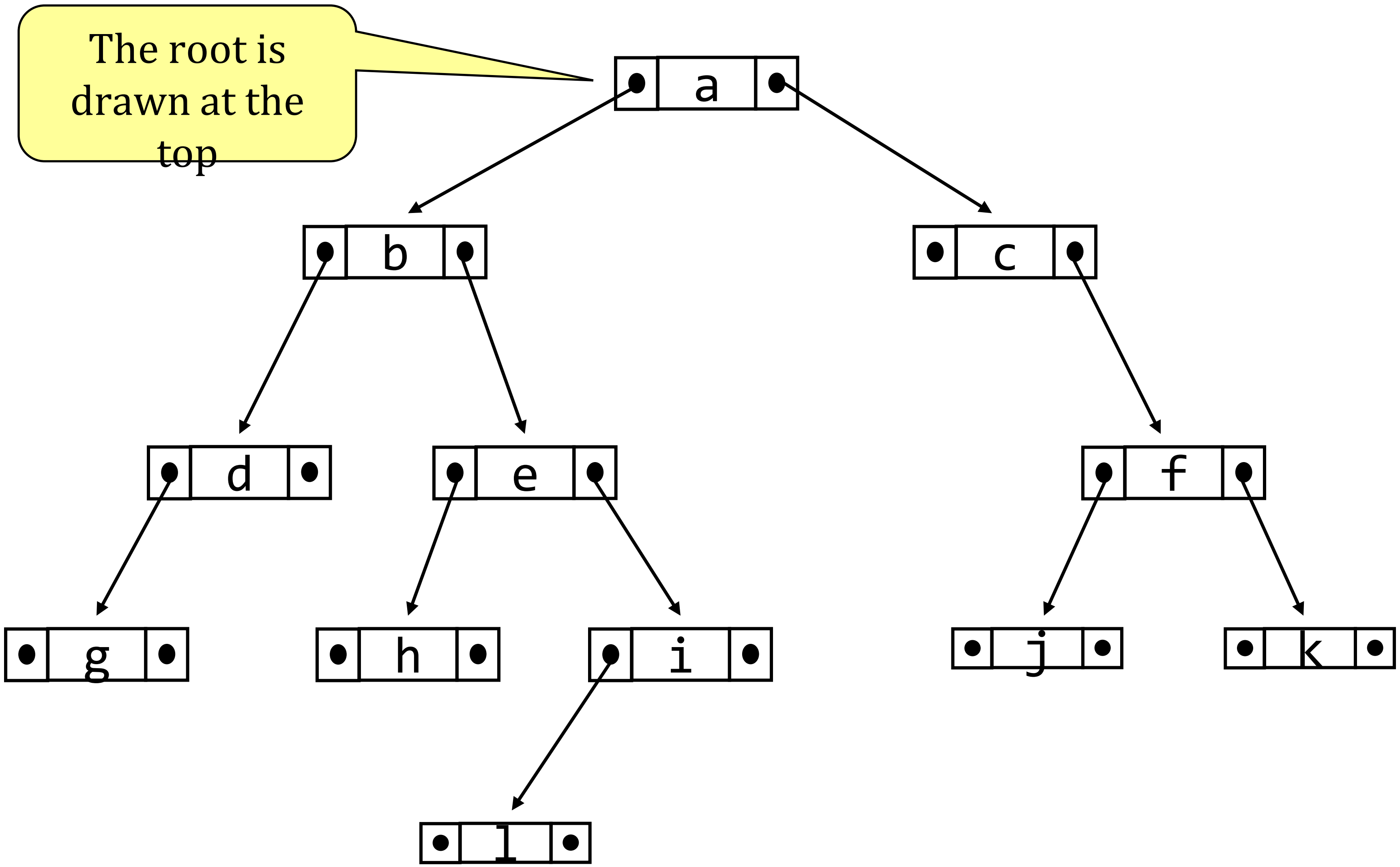
Binary Tree

- **Definition**: A *binary tree* is a rooted tree in which no vertex has more than two children





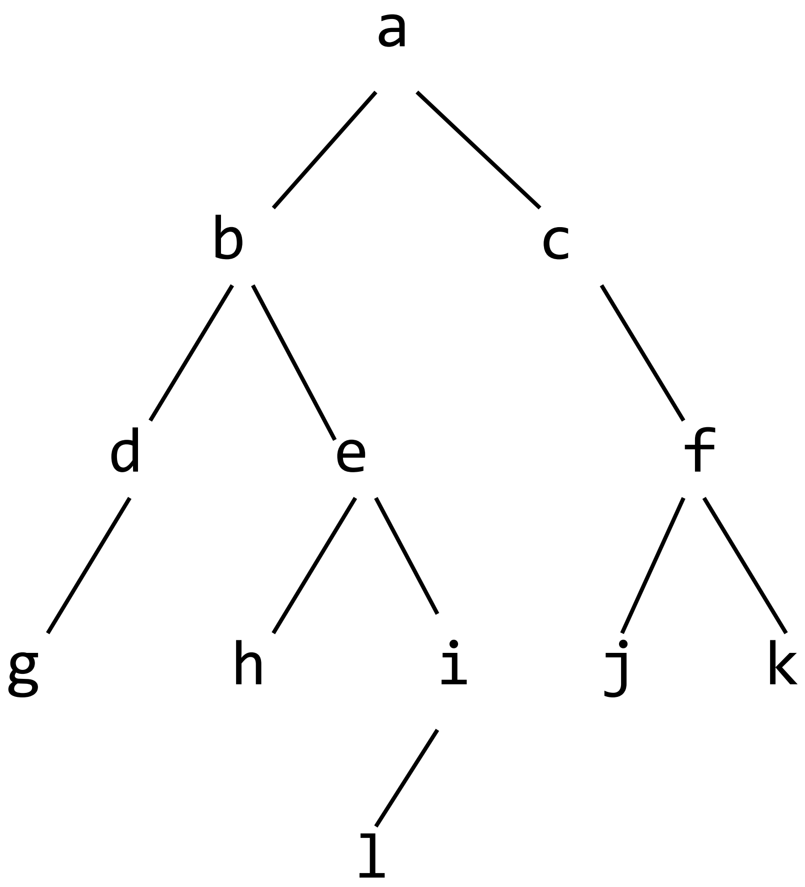
Binary Tree


The root is drawn at the top

Binary Tree

- Each node contains:
- A *value* (some sort of data item)
- A reference or pointer to a left child (may be null), and
- A reference or pointer to a right child (may be null)
- A binary tree may be empty (contain no nodes)
- If not empty, a binary tree has a *root* node
- Every node in the binary tree is reachable from the root node by a unique path

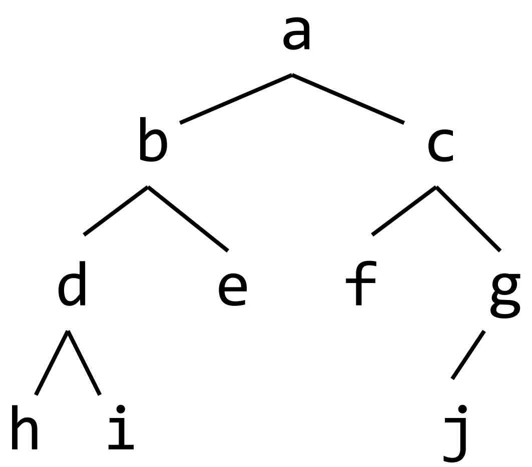
Binary Tree



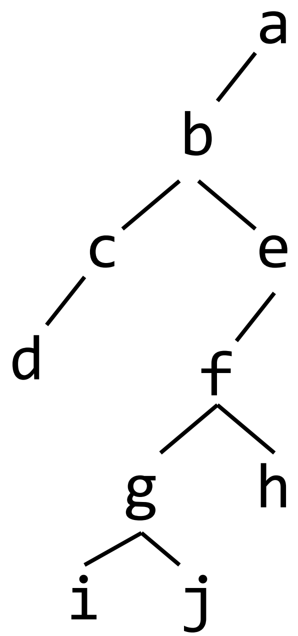
- size is 12
- a is at depth zero
- e is at depth 2
- The depth of a binary tree is the depth of its deepest node
- This tree has depth 4

Balance

- A binary tree is balanced if every level above the lowest is "full" (contains 2^n nodes)
- In most applications, a reasonably balanced binary tree is desirable



A balanced binary tree



An unbalanced binary tree

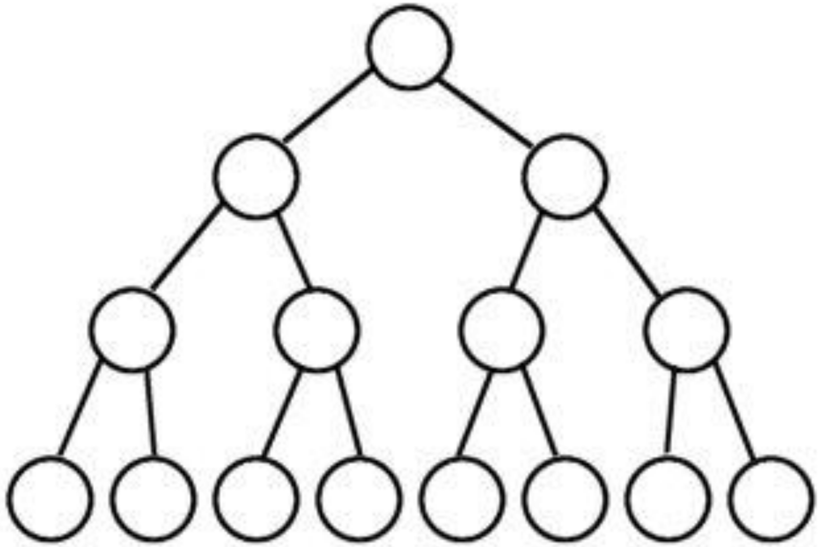
Tree traversals

- A binary tree is defined recursively: it consists of a *root*, a *left subtree*, and a *right subtree*
- To *traverse* (or walk) the binary tree is to visit each node in the binary tree exactly once
- Tree traversals are naturally recursive
- Inorder
- Preorder
- Postorder

Full Tree

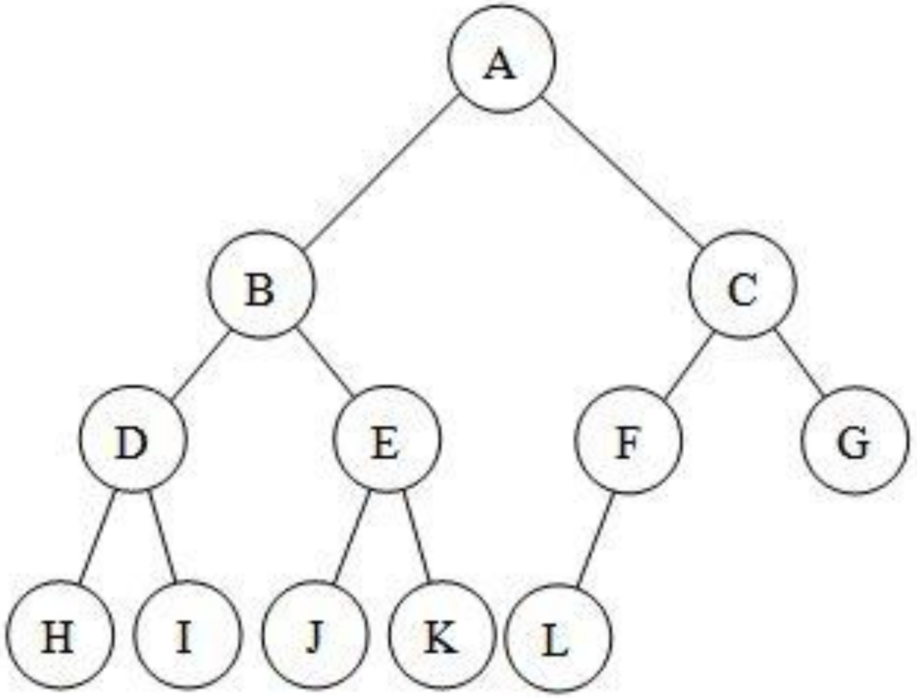
- A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children

Full Binary Tree

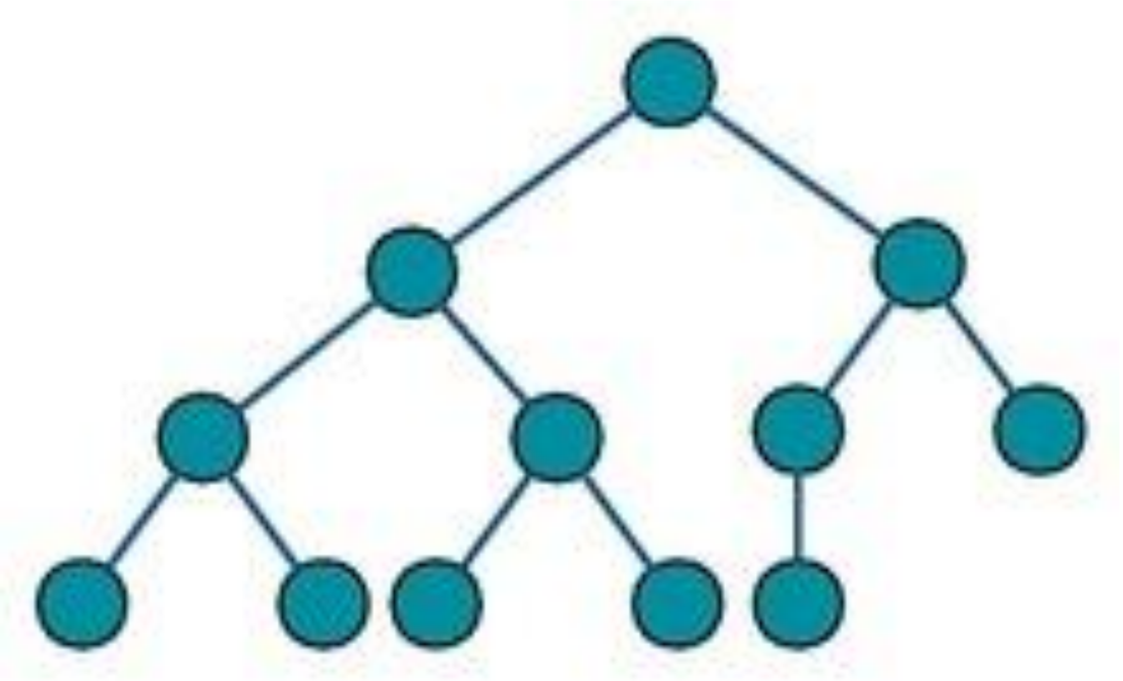


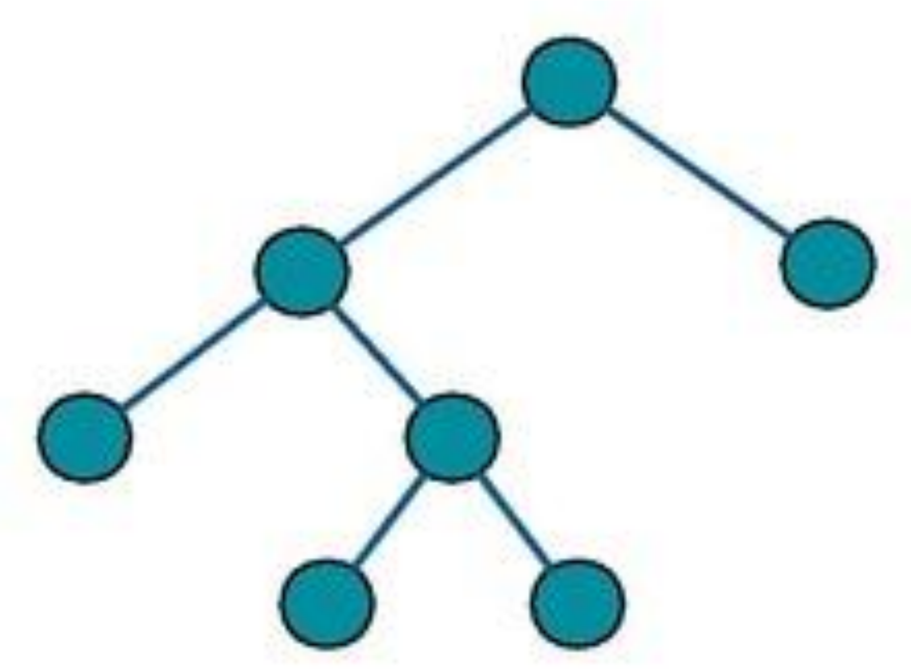
Complete Tree

- 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 Tree





A full but *not* complete binary tree

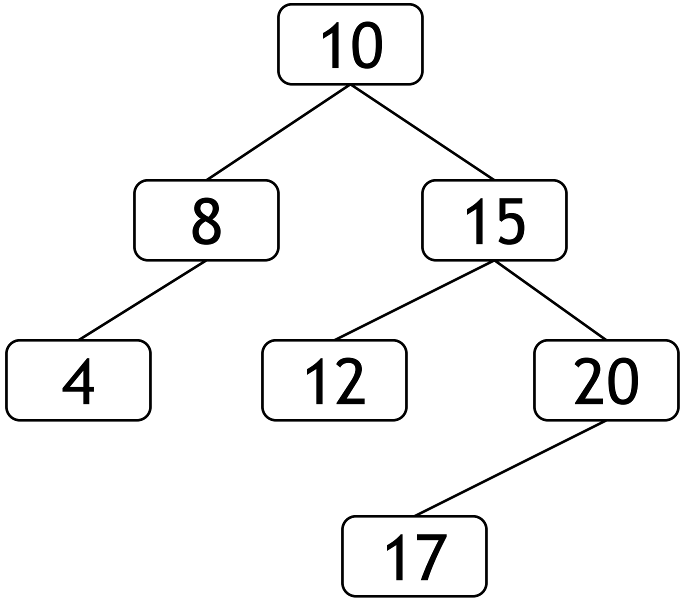
Binary search tree

A binary tree that has the following properties:

- The left subtree of a node contains only nodes with data less than the node's data.
- The right subtree of a node contains only nodes with data greater than the node's data.
- Both the left and right subtrees are also binary search trees.

Binary search tree

- Equal nodes can go either on the left or the right (but it has to be consistent)

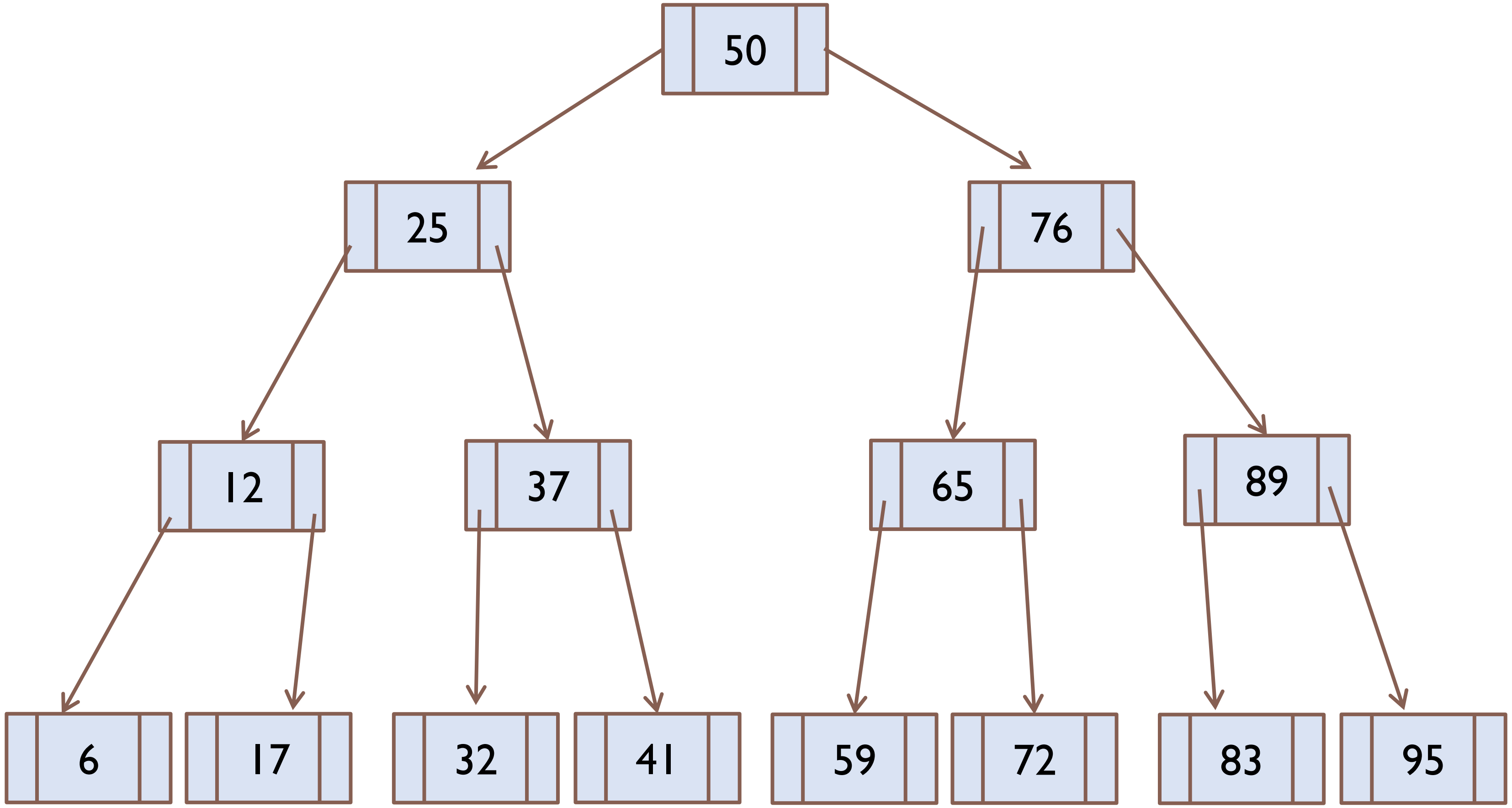


Binary search tree

Consequences:

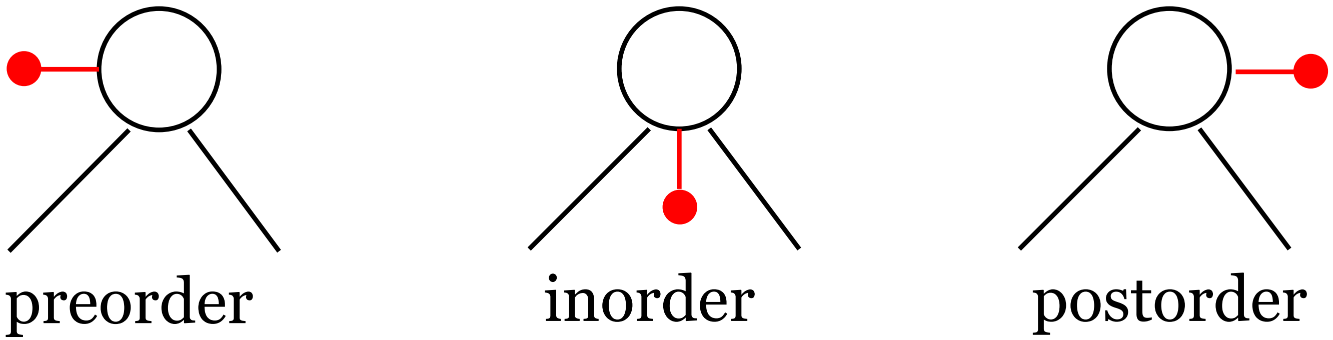
- The smallest element in a binary search tree (BST) is the “left-most” node
- The largest element in a BST is the “right-most” node
- Inorder traversal of a BST encounters nodes in *increasing* order

Binary search tree



Tree traversal

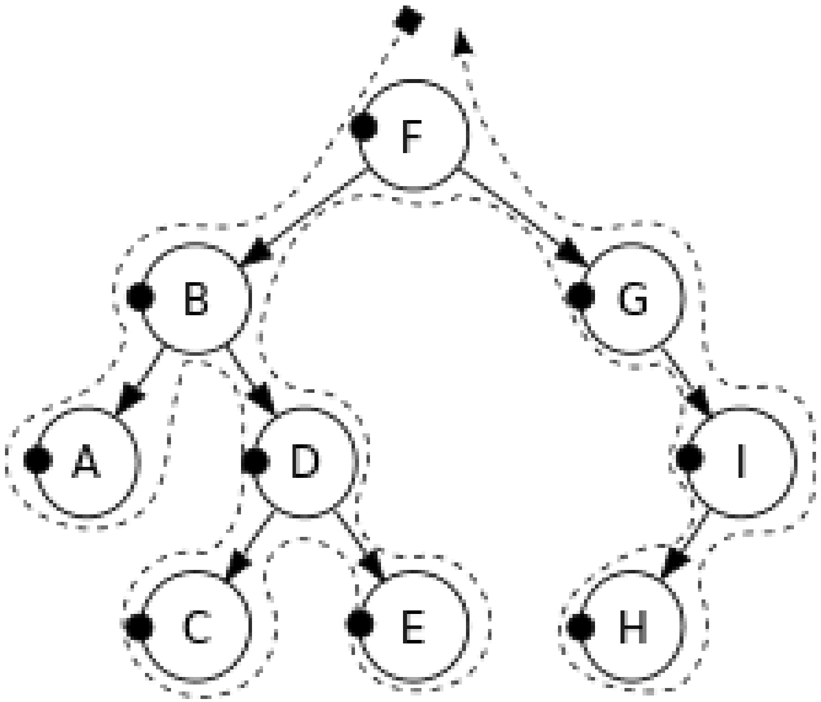
- The order in which the nodes are visited during a tree traversal can be easily determined by imagining there is a “flag” attached to each node, as follows:



Preorder traversal

In preorder, the root is visited first

- Check if the current node is empty / null
- Display the data part of the root (or current node)
- Traverse the left subtree by recursively calling the preorder function
- Traverse the right subtree by recursively calling the preorder function

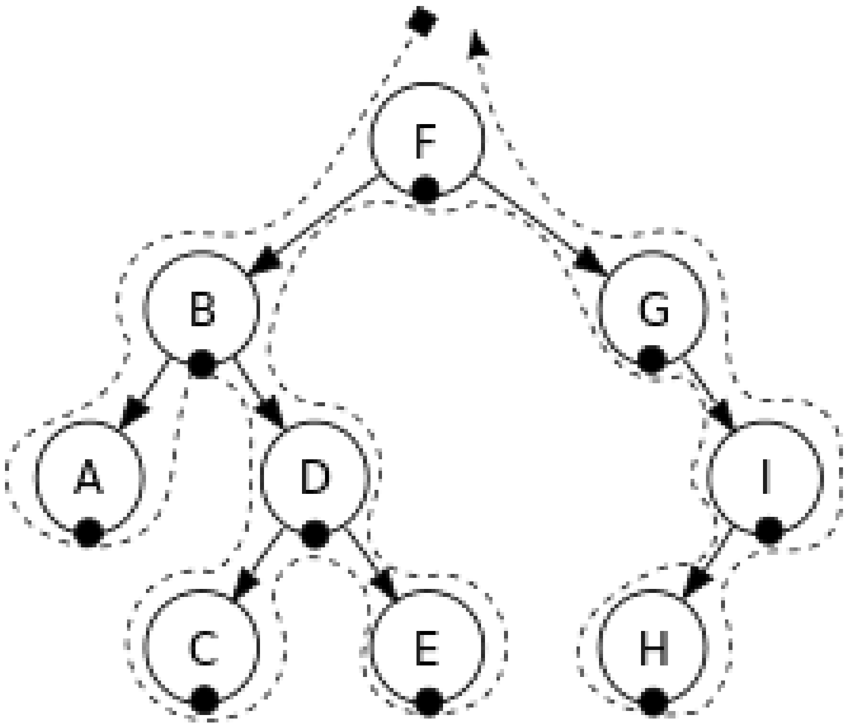


Pre-order: F, B, A, D, C, E, G, I, H.

Inorder traversal

In a search tree, in-order traversal retrieves data in sorted order

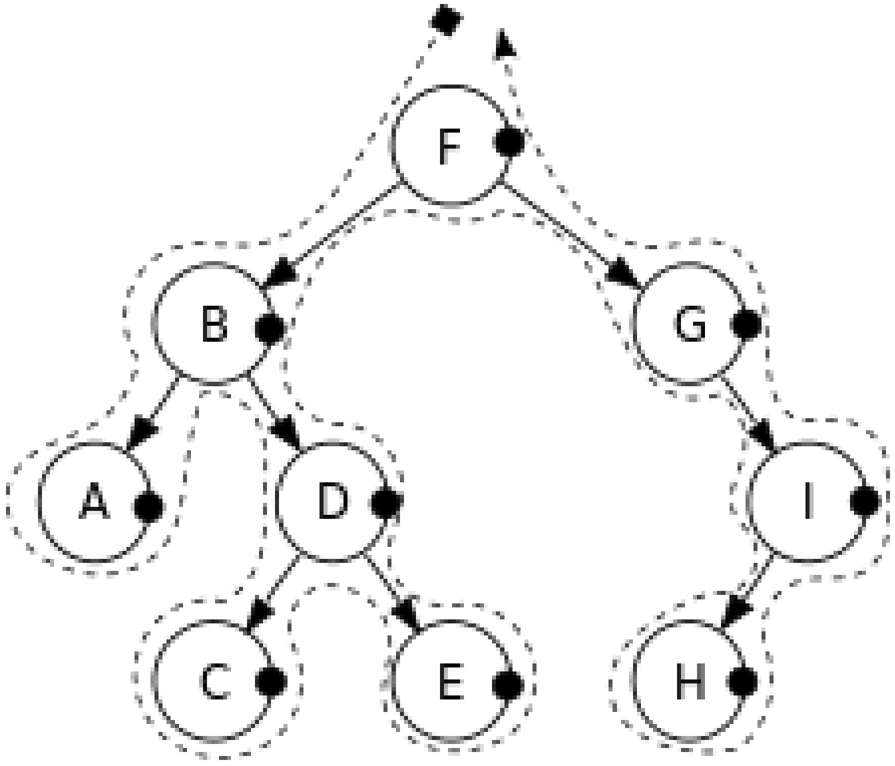
- Check if the current node is empty / null.
- Traverse the left subtree by recursively calling the in-order function.
- Display the data part of the root (or current node).
- Traverse the right subtree by recursively calling the in-order function



In-order: A, B, C, D, E, F, G, H, I

Postorder traversal

- Check if the current node is empty / null.
- Traverse the left subtree by recursively calling the post-order function.
- Traverse the right subtree by recursively calling the post-order function.
- Display the data part of the root (or current node)



In-order: A, C, E, D, B, H, I, G, F

Binary search tree

A binary tree can also be stored in arrays

If a node has an index i

- its children are found at indices $2i+1$ (for the left child) and $2i+2$ (for the right)
- its parent (if any) is found at index $\left\lfloor \frac{i-1}{2} \right\rfloor$