Data Structures

Lecture 17: Binary Trees

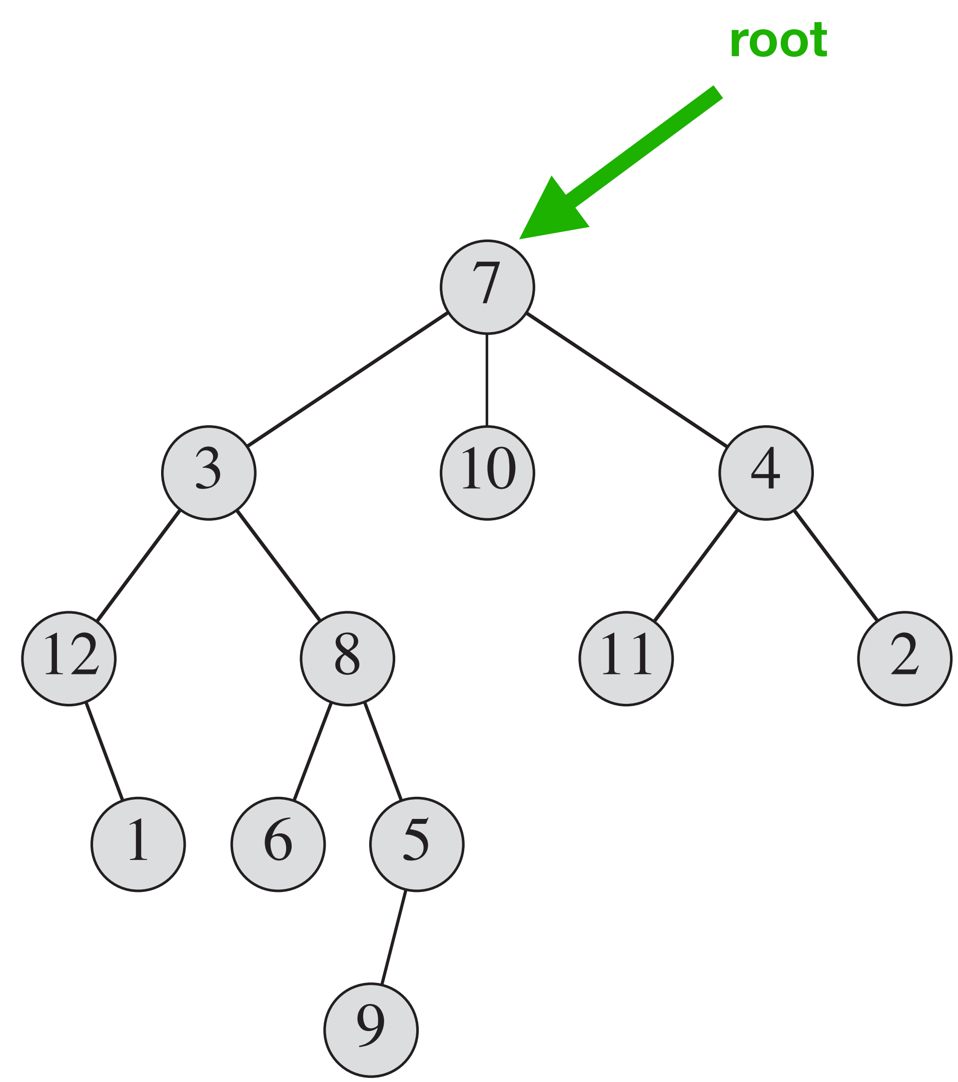
Nopadon Juneam
Department of Computer Science
Kasetsart university

Outlines

- Binary trees: basic terminology and notations
- Properties of binary trees
- Data structures for representing binary trees
  - Linked structure
  - Array-based structure

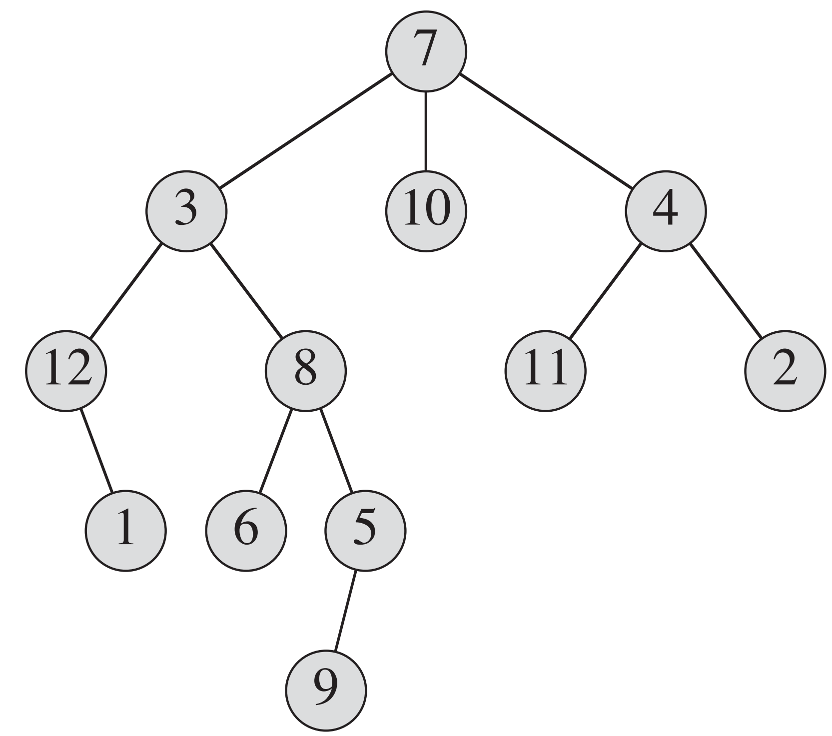
Rooted Trees

- A ***rooted tree*** is a free tree in which one of the vertices is distinguished from the others
  - We call the distinguished vertex the ***root*** of the tree (the top element of the tree)
  - We often refer to a vertex of a rooted tree as a ***node*** of the tree



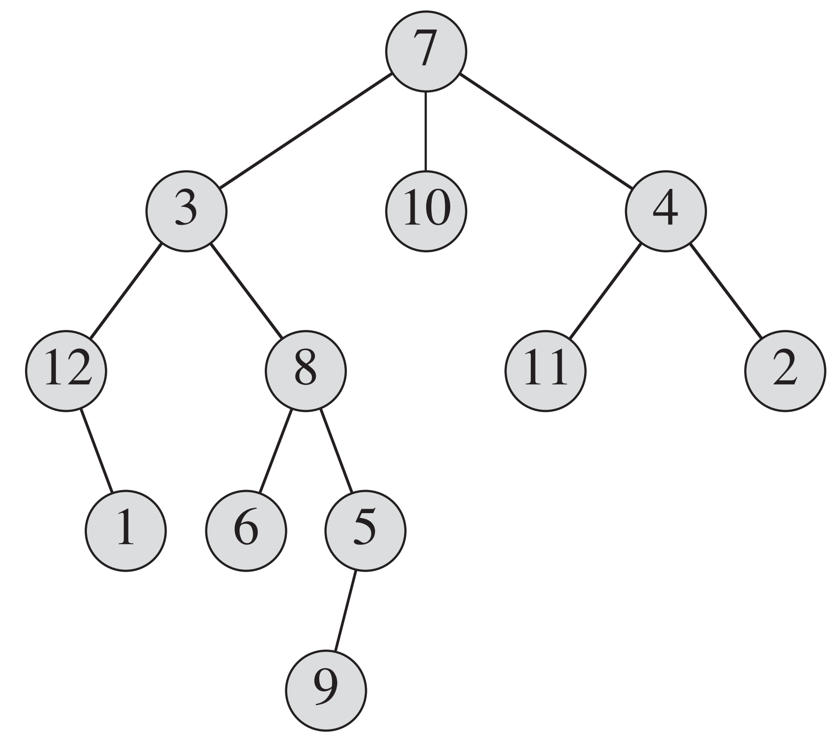
Rooted Tree Terminology (1)

- Consider a node *x* in a rooted tree *T* with root *r*:
  - We call *any* node *y* on the unique simple path from *r* to *x* an ***ancestor*** of *x*
  - If *y* is an ancestor of *x*, then *x* is a ***descendant*** of *y* (every node is both an ancestor and a descendant of itself)
  - The ***subtree rooted at x*** is the tree induced by descendants of *x*, rooted at *x*
    - For example, the subtree rooted at node 8 in the figure contains nodes 8, 6, 5, and 9



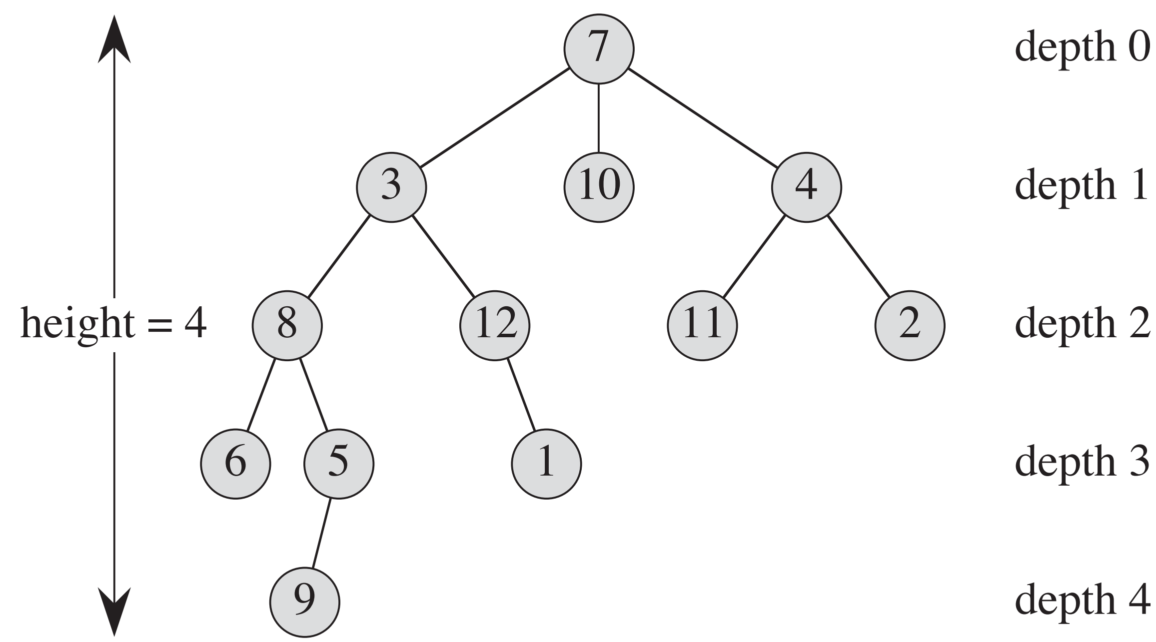
Rooted Tree Terminology (2)

- If the last edge on the simple path from the root *r* of a tree *T* to a node *x* is (*y*, *x*), then *y* is the ***parent*** of *x*, and x is a ***child*** of *y*
    - The root is the only node in *T* with no parent
- If two nodes have the same parent, they are ***siblings***
- A node with no children is a ***leaf*** or ***external node***
- A non-leaf node is an ***internal node***

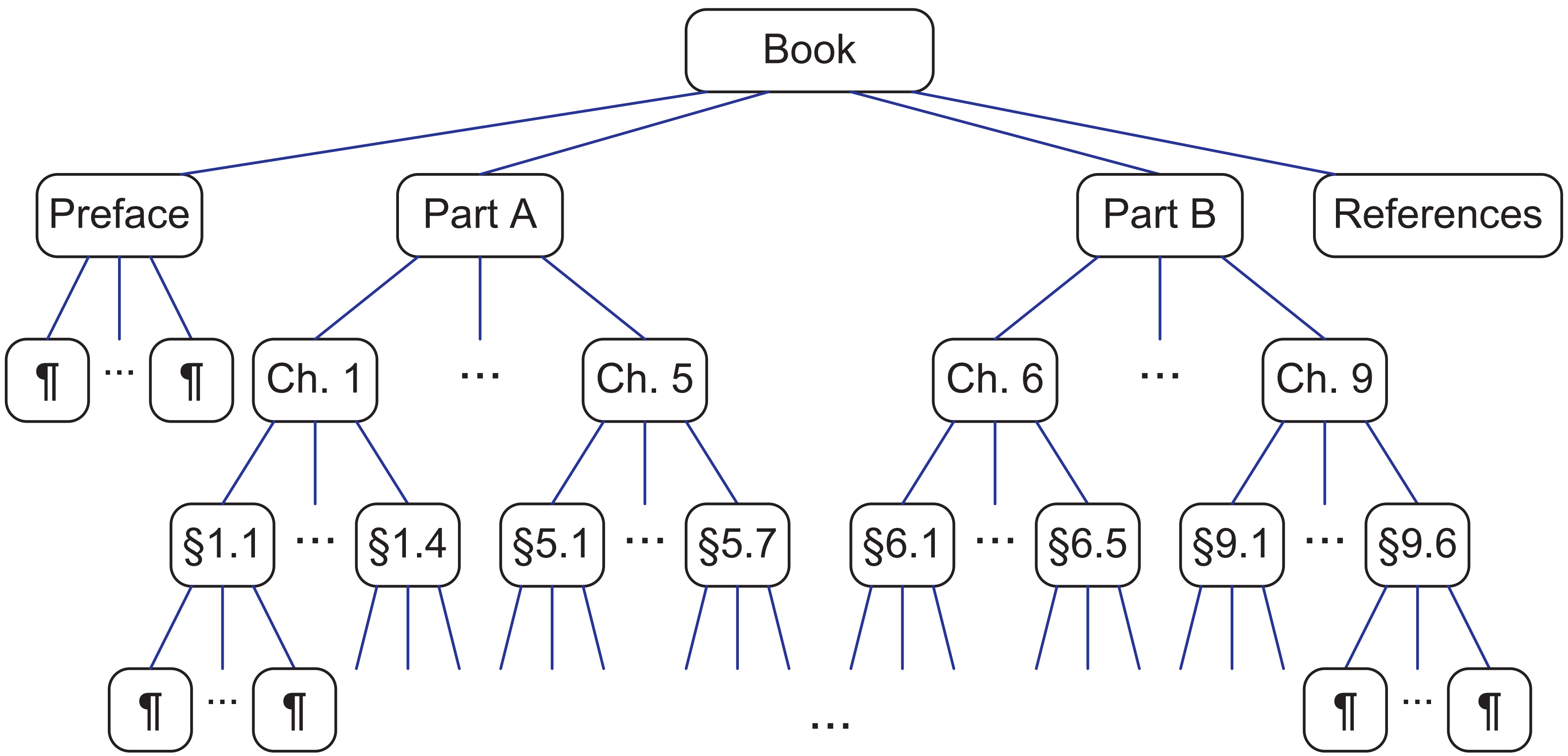


Rooted Tree Terminology (3)

- The number of children of a node *x* in a rooted tree *T* equals the ***degree*** of *x*
- The length of the simple path from the root *r* to a node *x* is the ***depth*** of *x* in *T*
- A ***level*** of a tree consists of all nodes at the same depth.
- The ***height*** of a node in a tree is the number of edges on the longest simple downward path from the node to a leaf, and the height of a tree is the height of its root
    - The height of a tree is also equal to the largest depth of any node in the tree.

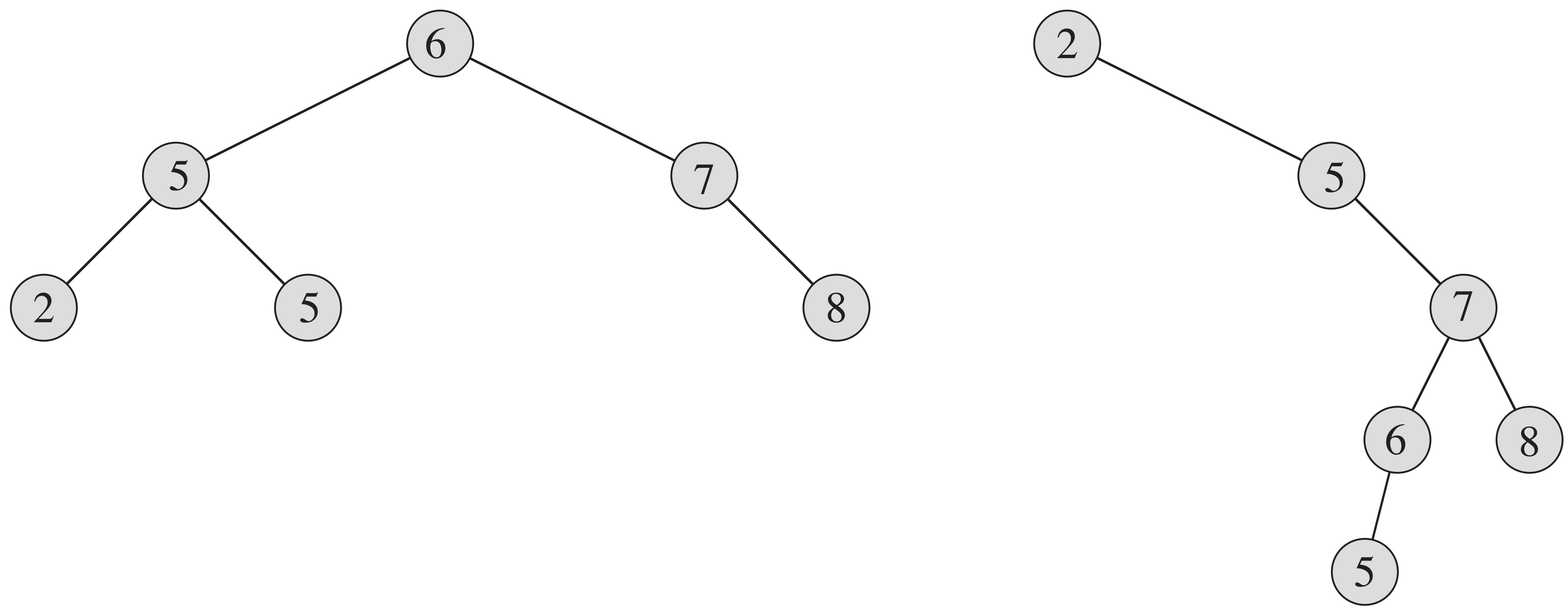


Ordered Trees



- An ***ordered tree*** is a rooted tree in which the children of each node are ordered. That is, if a node has *k* children, then there is a first child, a second child, . . . , and a *k*-th child

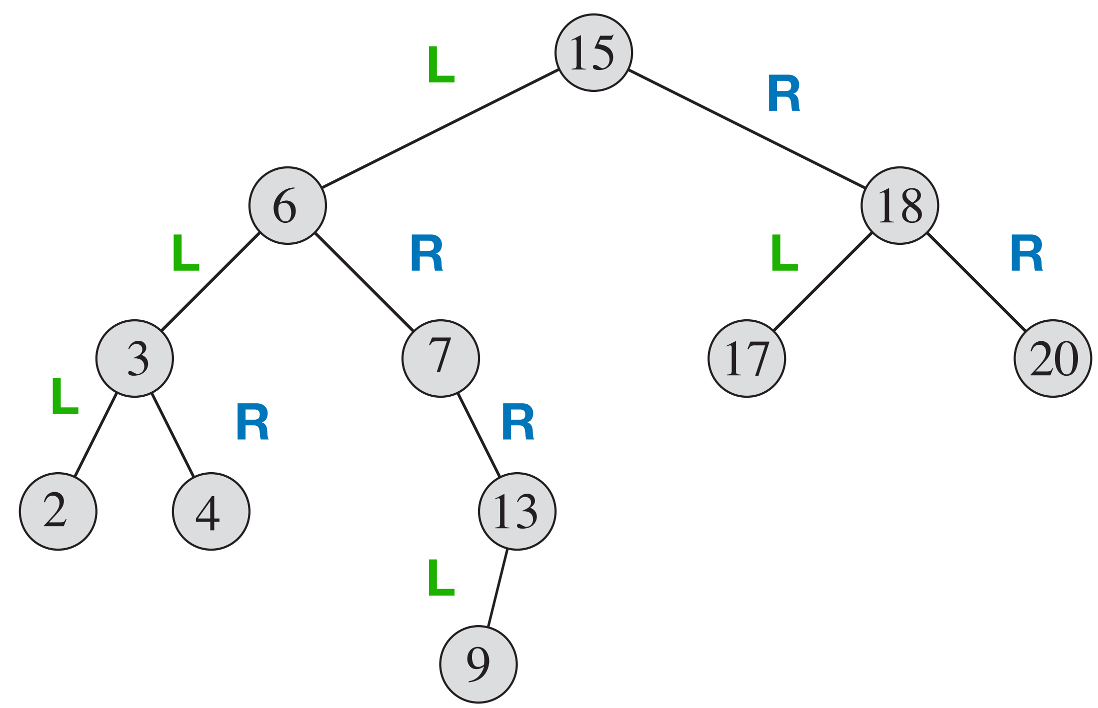
Binary Trees



- A ***binary tree*** is kind of an ordered tree in which every node has at most two children. However, if a node has just one child, the position of the child matters

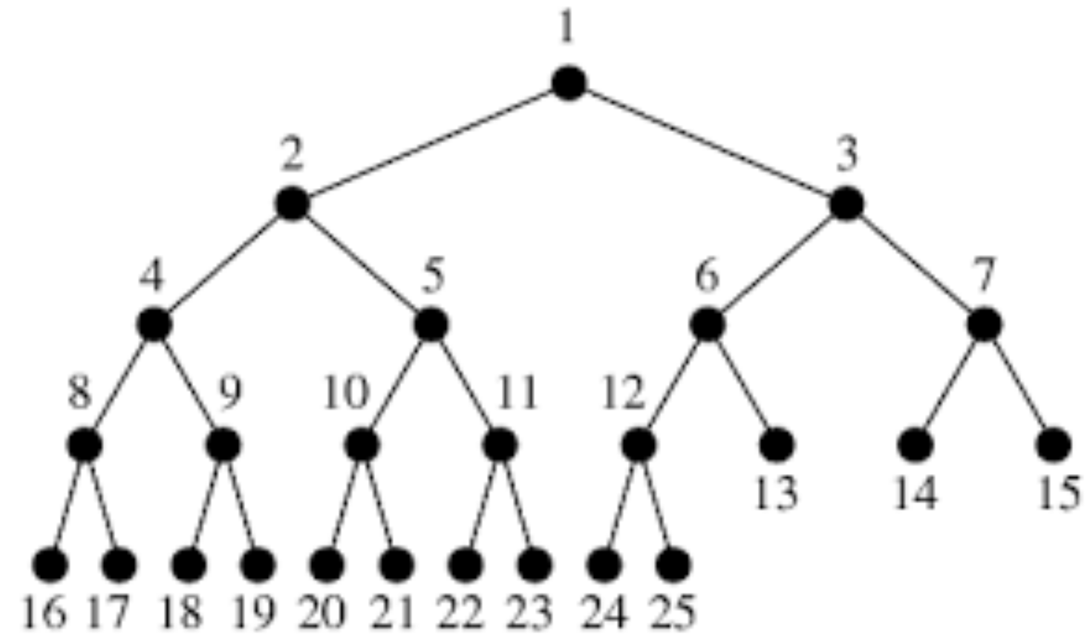
Binary Tree Terminology (1)

- In a binary tree, every child node is labeled as being either a ***left child*** or a ***right child***
  - A left child *precedes* a right child in the ordering of children of a node
- The subtree rooted at a left or right child of an internal node is called the node's ***left subtree*** or ***right subtree***, respectively

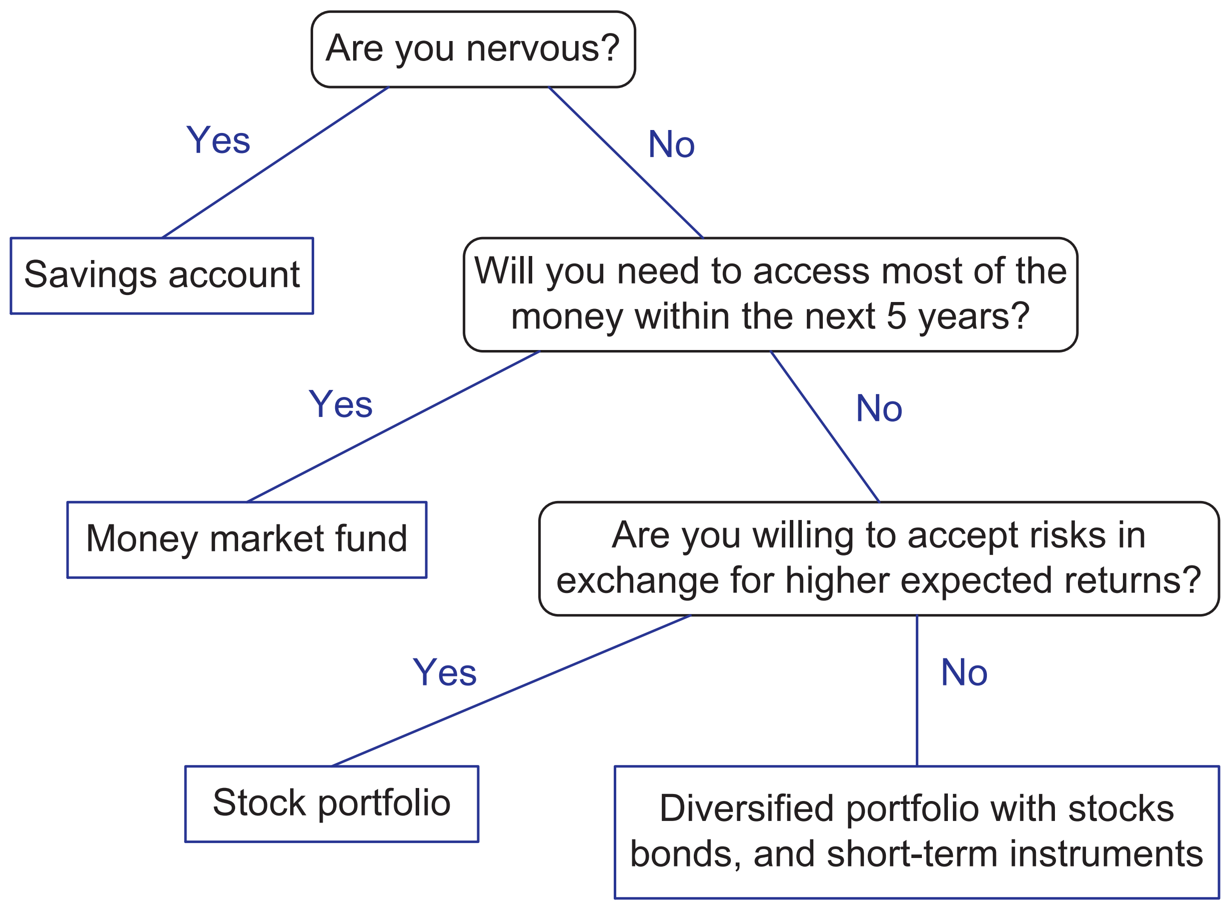


Binary Tree Terminology (2)

- A binary tree is ***proper*** if each node has either zero or two children
  - Some people also refer to such trees as being ***full*** binary trees
  - Thus, in a proper binary tree, every internal node has exactly two children. A binary tree that is not proper is ***improper***



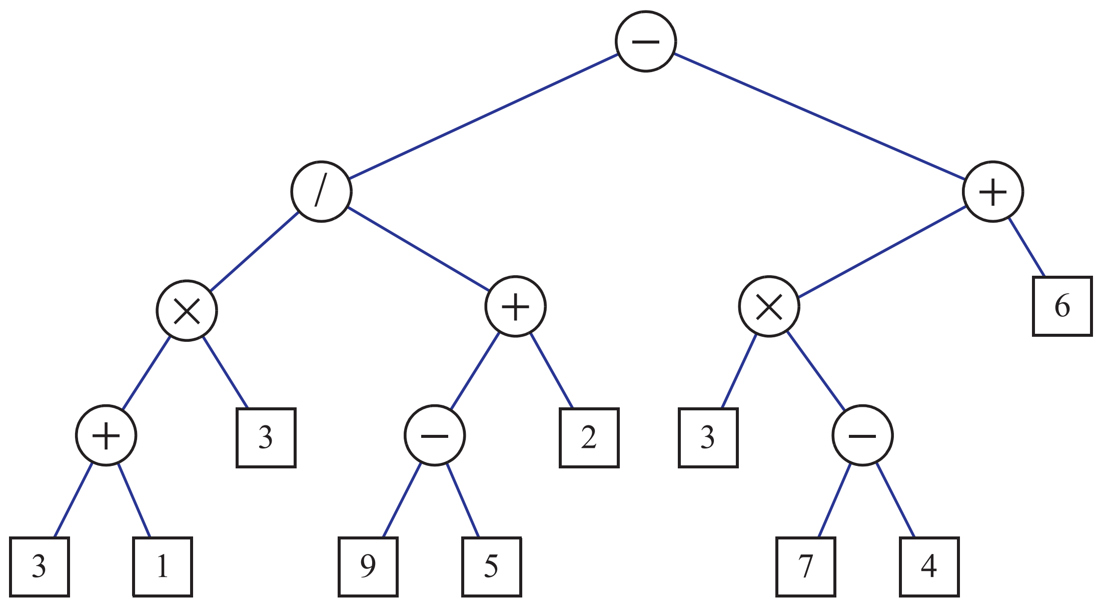
Decision Trees: Class of Binary Trees



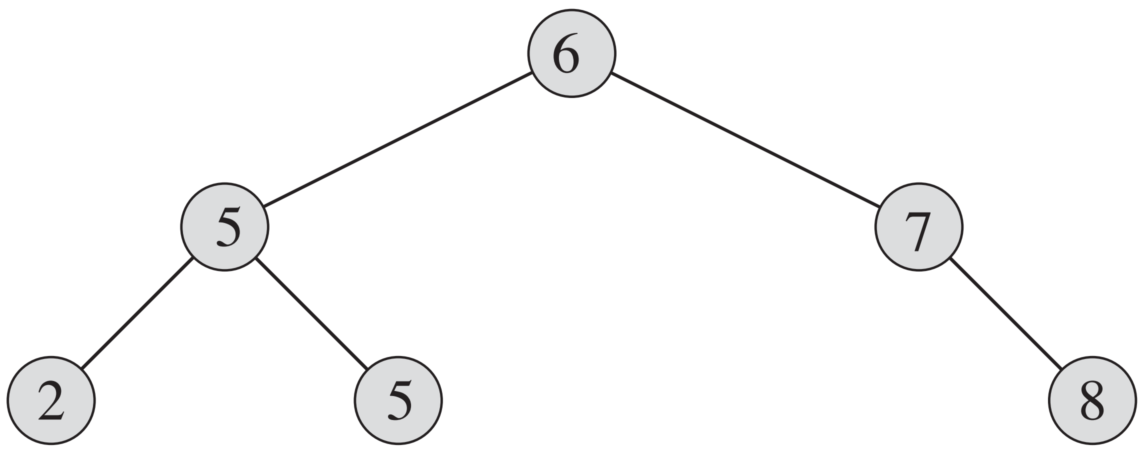
- An important class of binary trees called **decision trees** arises in contexts where we wish to represent a number of different outcomes that can result from answering a series of yes-or-no questions

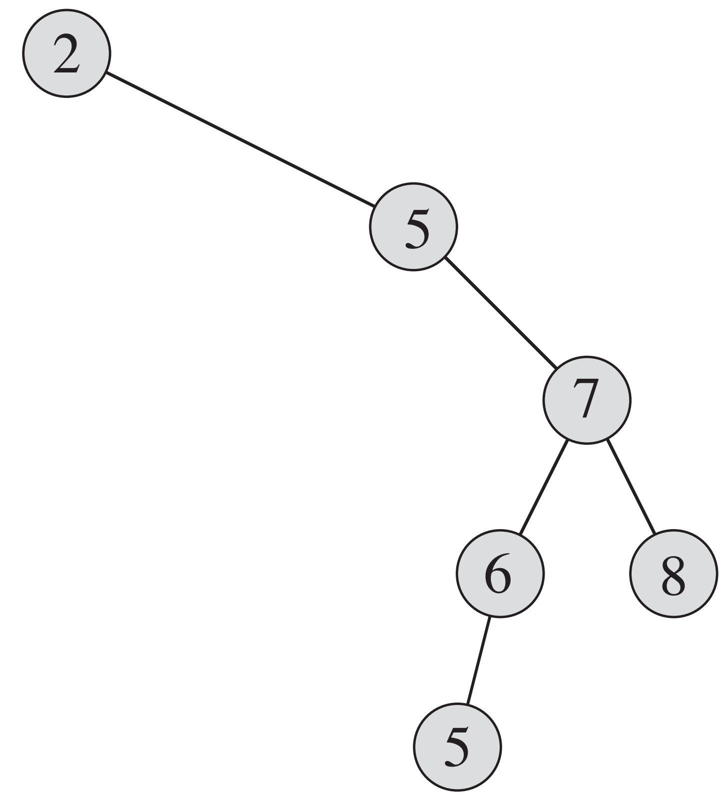
Applications of Binary Trees

- A binary tree can be used to represent an *arithmetic expression:*
    - Each node in such a tree has a value associated with it
    - If a node is external, then its value is that of its variable or constant
    - If a node is internal, then its value is defined by applying its operation to the values of its children
- The above tree represents the expression $((((3+1)\times3)/((9-5)+2))-((3\times(7-4))+6))$
- The value associated with the internal node labeled "/" is 2



Binary Trees (Recursive Definition)

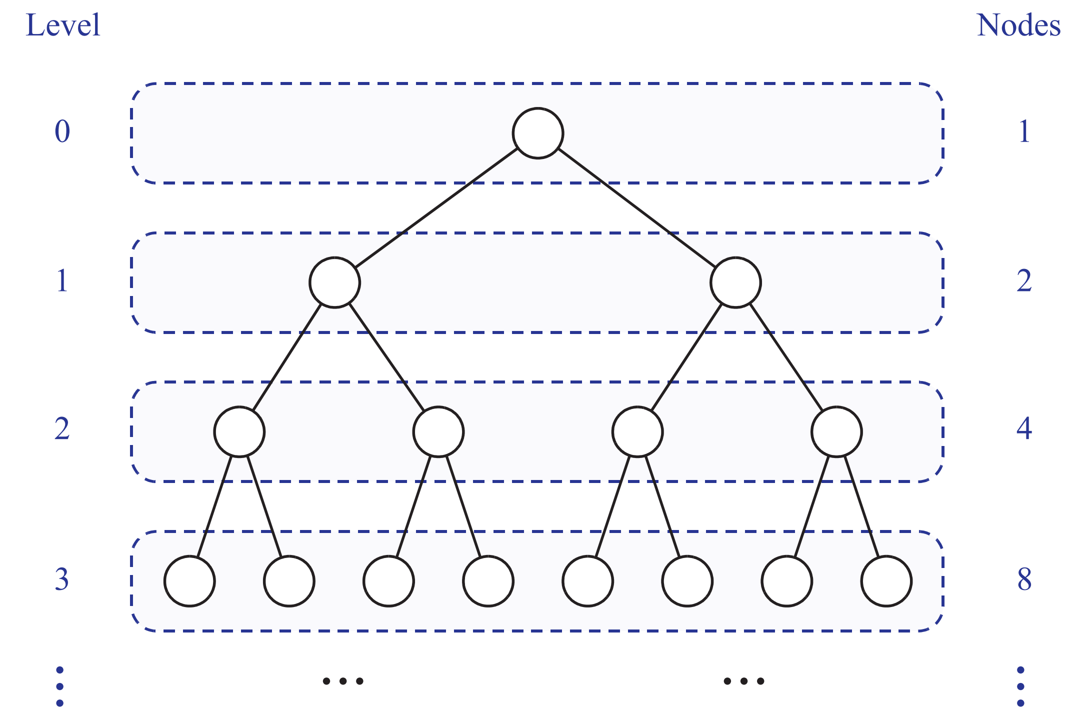




- A ***binary tree*** *T* is either empty or consists of:
  - A node r, called the ***root*** of *T* and storing an element
  - A binary tree, called the ***left subtree*** of *T*
  - A binary tree, called the ***right subtree*** of *T*

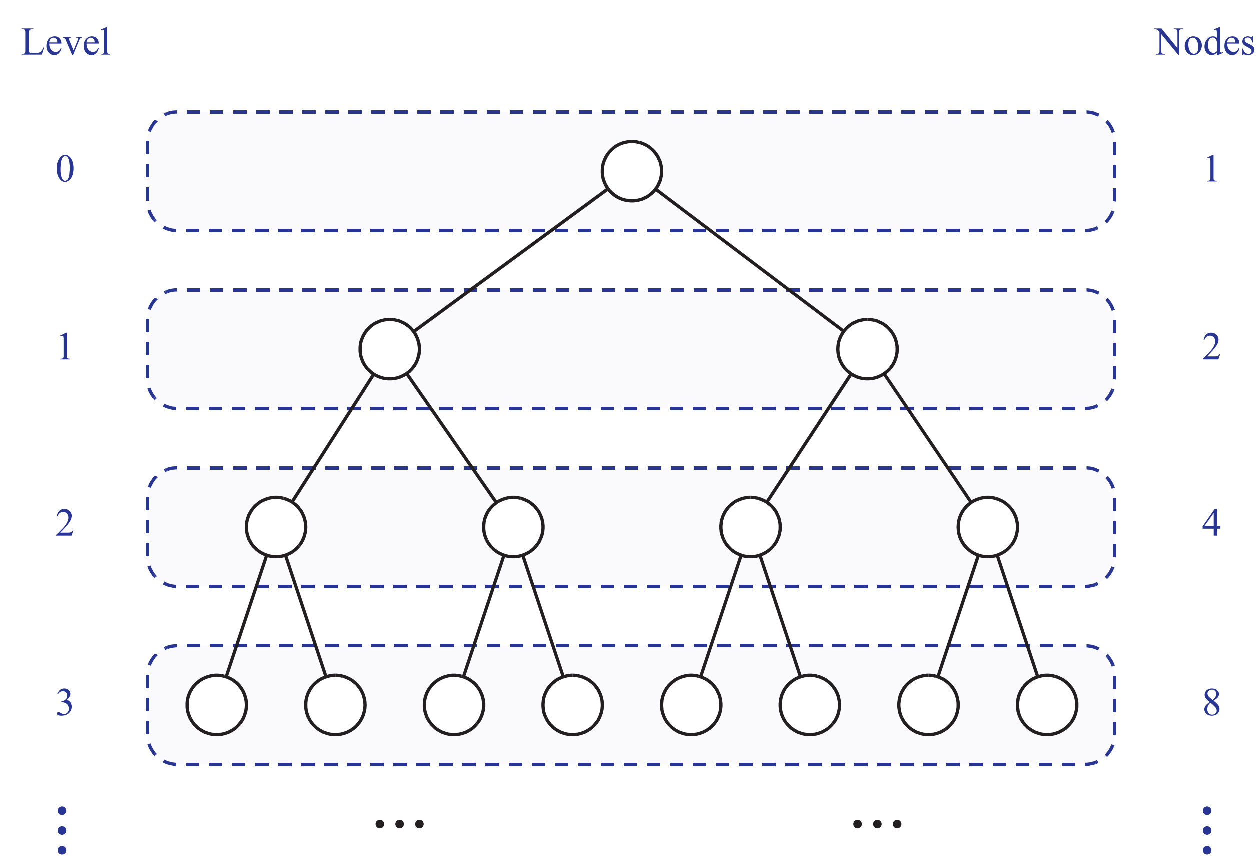
Binary Trees's Properties (1)

- Binary trees have several interesting properties dealing with relationships between their *heights* and *number of nodes*:
  - We denote the set of all nodes of a tree T, at the same depth *d*, as the ***level*** *d* of *T:*
    - Level 0 has one node (the root)
    - Level 1 has at most two nodes (the children of the root)
    - Level 2 has at most four nodes, and so on.
    - In general, level d has at most 2^d nodes
- *Remark*: The maximum number of nodes on the levels of a binary tree grows exponentially as we go down the tree



Binary Trees's Properties (2)

- **Proposition 1**: Let T be a nonempty binary tree. Let n, n_E , n_I and h denote the number of nodes, number of external nodes, number of internal nodes, and height of T, respectively. Then T has the following properties:

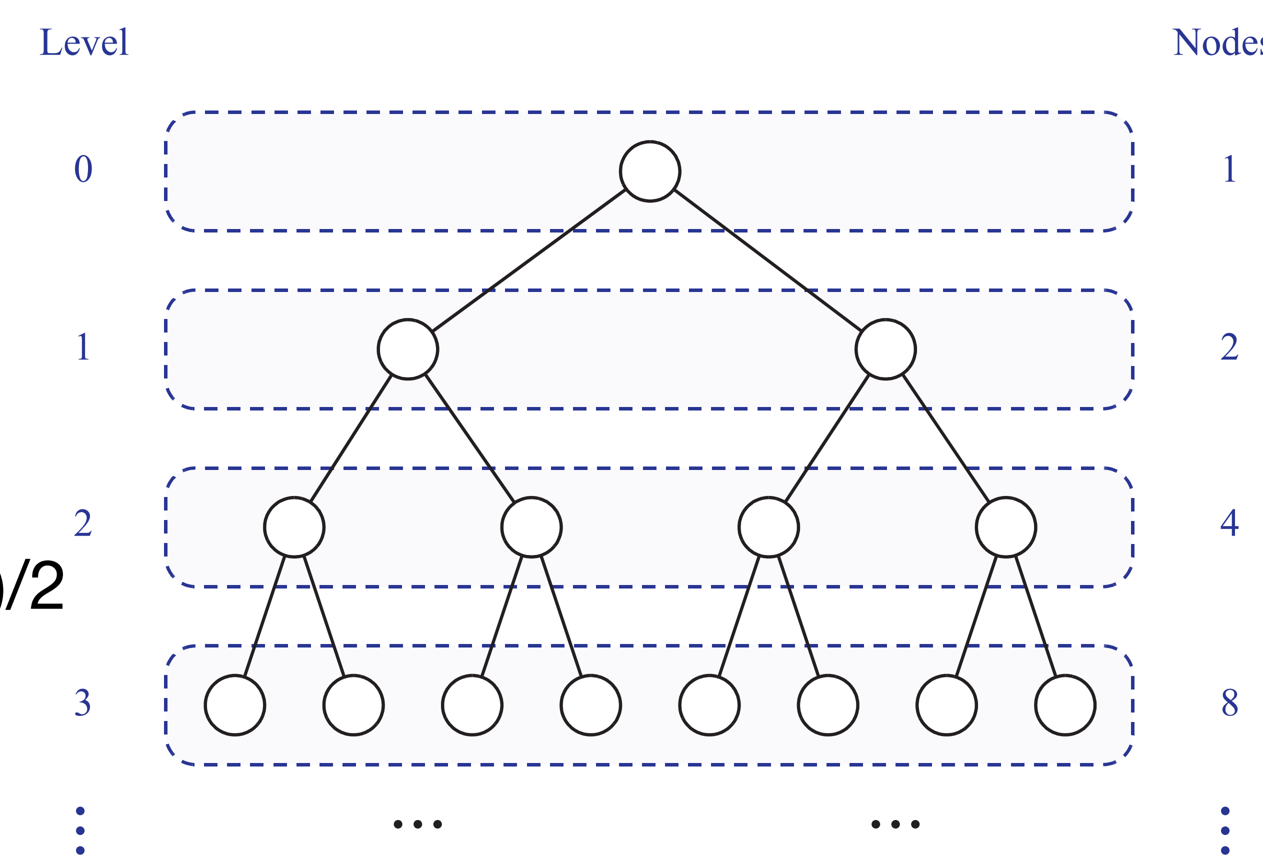


1. $h+1 \le n \le 2^{h+1}-1$
2. $1 \le n_E \le 2^h$
3. $h \le n_I \le 2^h - 1$
4. $\log(n+1)-1 \le h \le n-1$

Binary Trees's Properties (3)

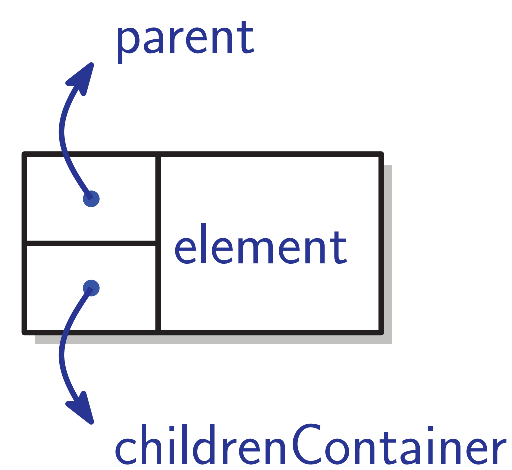
- **Proposition 2**: Let T be a *proper* binary tree. Let n, n_E , n_I and h denote the number of nodes, number of external nodes, number of internal nodes, and height of T, respectively. Then T has the following properties:

1. $2h+1 \leq n \leq 2^{h+1}-1$
2. $h+1 \leq n_E \leq 2^h$
3. $h \leq n_I \leq 2^h - 1$
4. $\log(n+1)-1 \leq h \leq (n-1)/2$

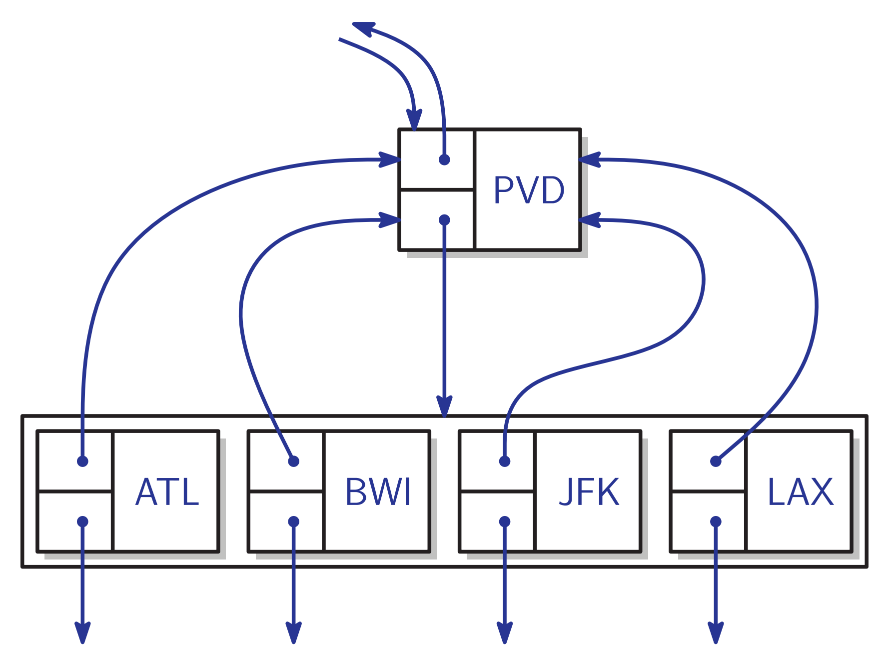


Linked Structure for General Trees

- A natural way to realize a tree T is to use a ***linked structure***, where we represent each node of T by a n object p with the following fields:



  - A reference to the node's element
  - A link to the node's parent
  - Some kind of collection (for example, a list or array) to store links to the node's children



Linked Structure for Binary Trees

- In a linked structure for a binary tree *T*, we represent each node of *T* by a node object *p* with the following fields:

  - A reference to the node's element

  - A link to the node's parent

  - A link to the node's two children