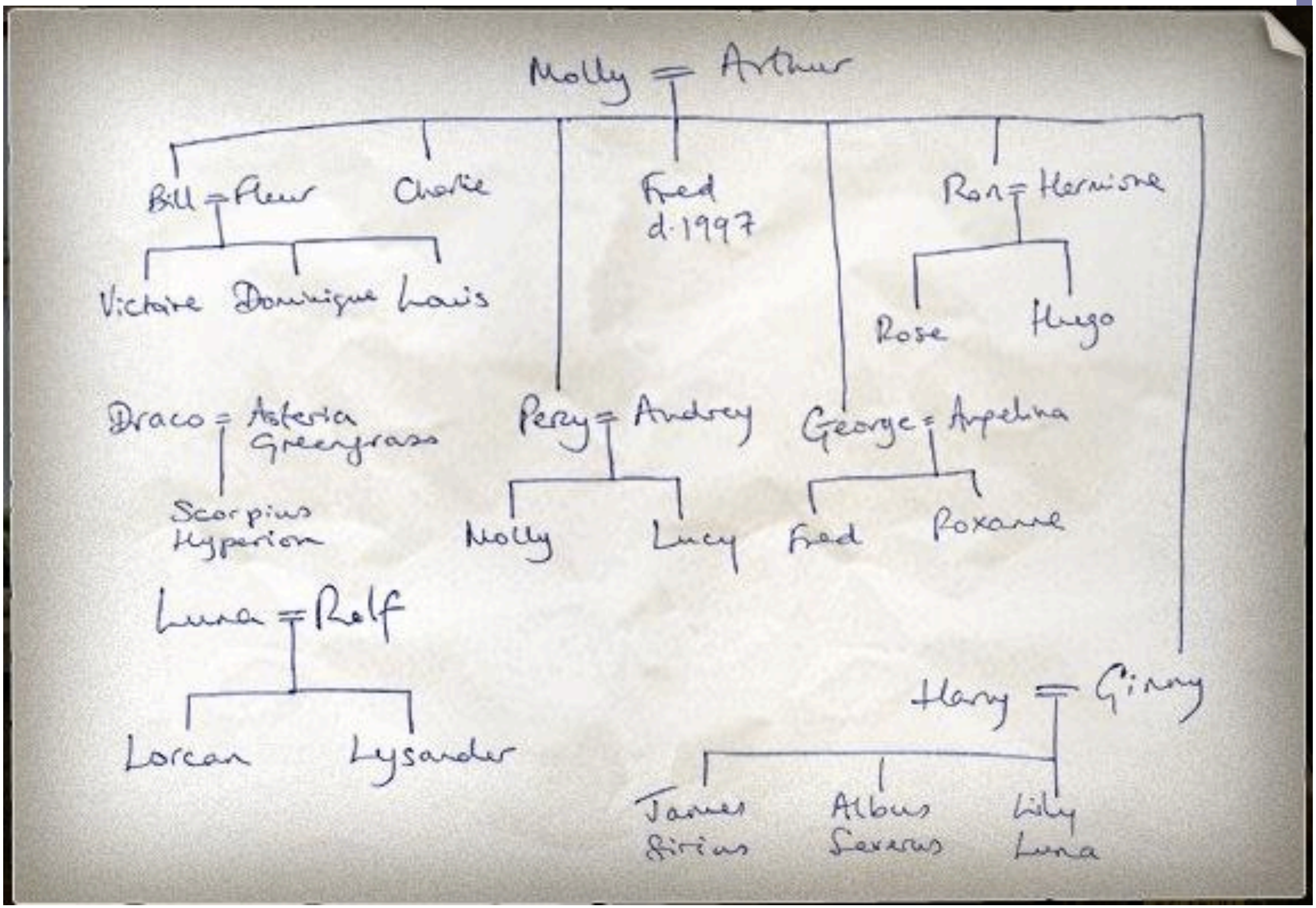
Binary Trees

Trees – hierarchical data structure



Trees in CS are different

- Nodes and links

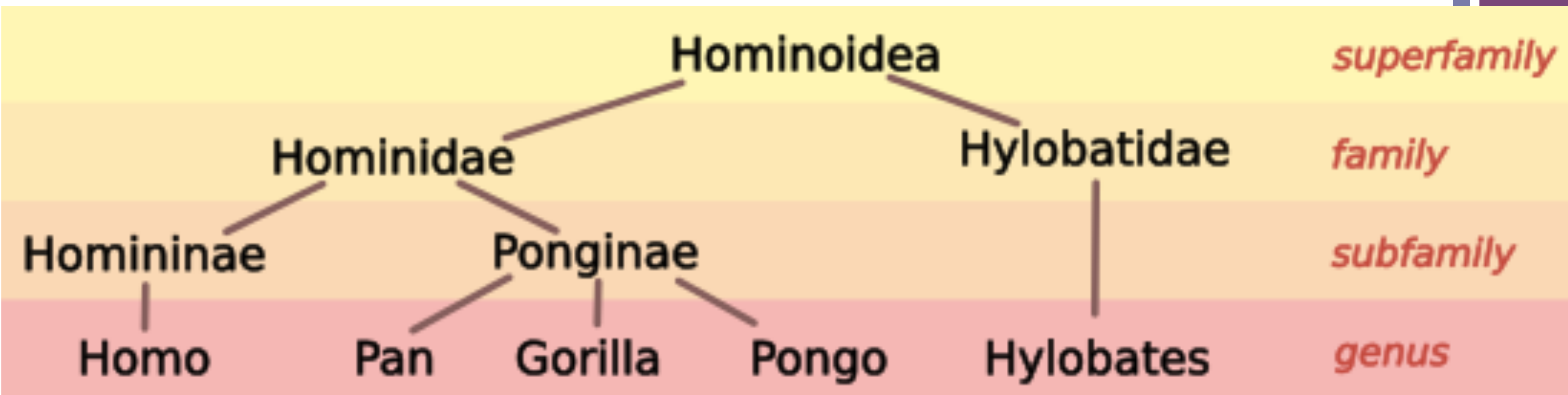


Image Source: https://en.wikipedia.org/wiki/Ape

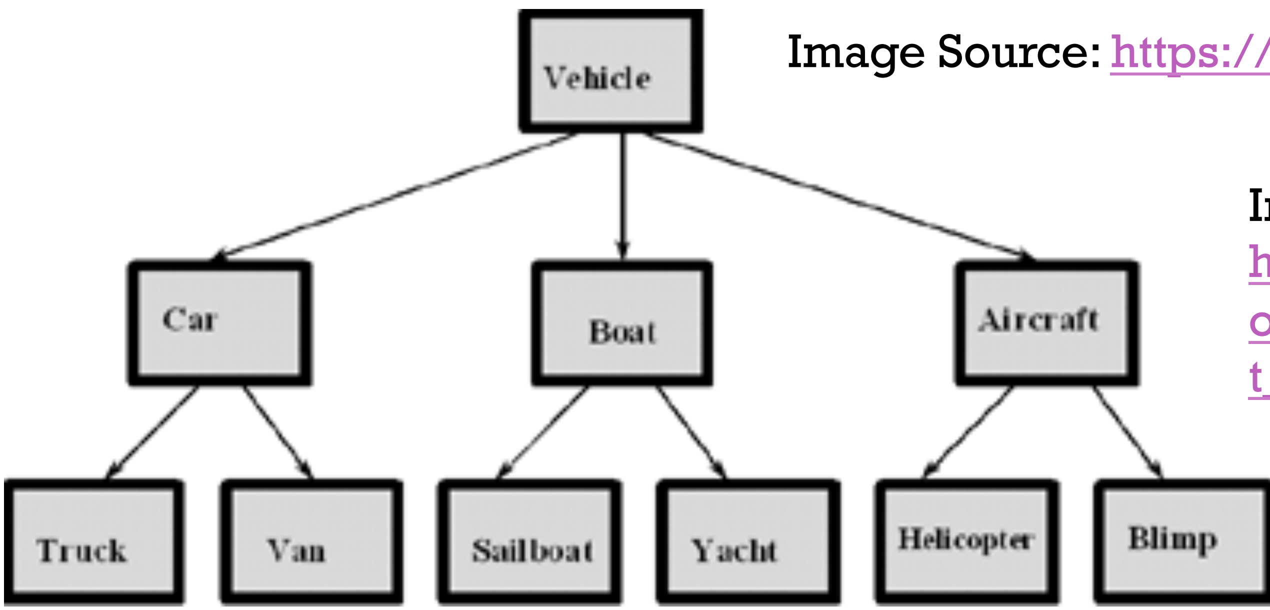
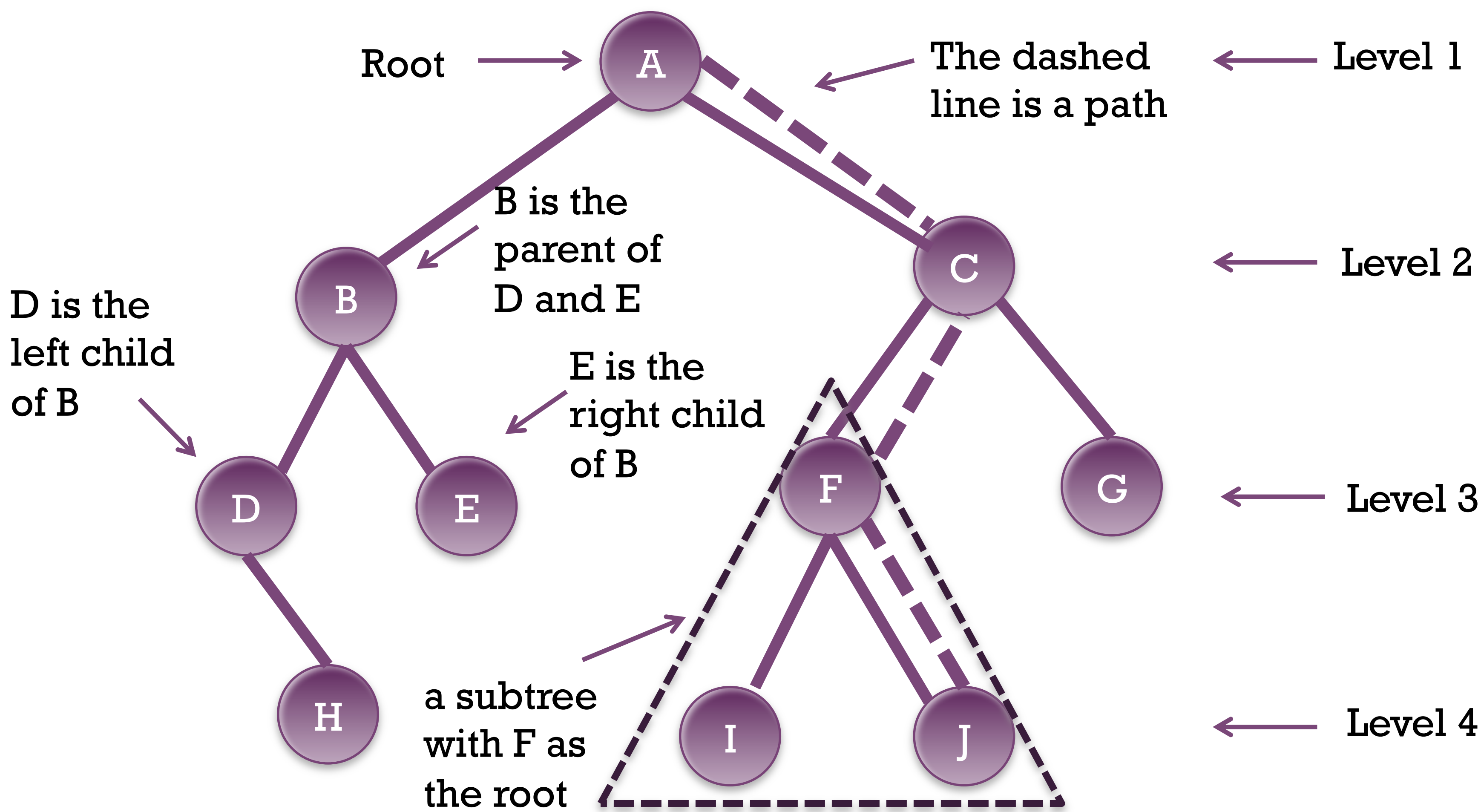


Image Source: http://www.dba-oracle.com/images/t_obje3.gif

Tree Terms


Root
A
The dashed
line is a path
Level 1
B is the
parent of
D and E
B
C
Level 2
D is the
left child
of B
E is the
right child
of B
D
E
F
G
Level 3
H
a subtree
with F as
the root
I
J
Level 4

Tree Properties

- each node has exactly one Parent
- leaf nodes have no children
- Level is distance from root:

$$\boldsymbol{level(node) = \begin{cases} 1\ if\ node\ is\ root \\ 1 + level\big(parent(node)\big)o/w \end{cases}}$$

- Height is # of nodes in largest path from root to leaf

$$\boldsymbol{height(tree) = \begin{cases} 0\ if\ tree\ is\ empty \\ 1 + max(height(c1), \ldots height(cn)) \end{cases}}$$

Binary Trees

- **Each node has at most two subtrees.**

  T is a Binary Tree if either one of the following is true:
  Definition:
  (1) T is empty.
  (2) If T is not empty, its root has two subtrees
  T_L and T_R such that T_L and T_R are Binary Trees.

- **Let's look at some examples**

Types of Binary Trees

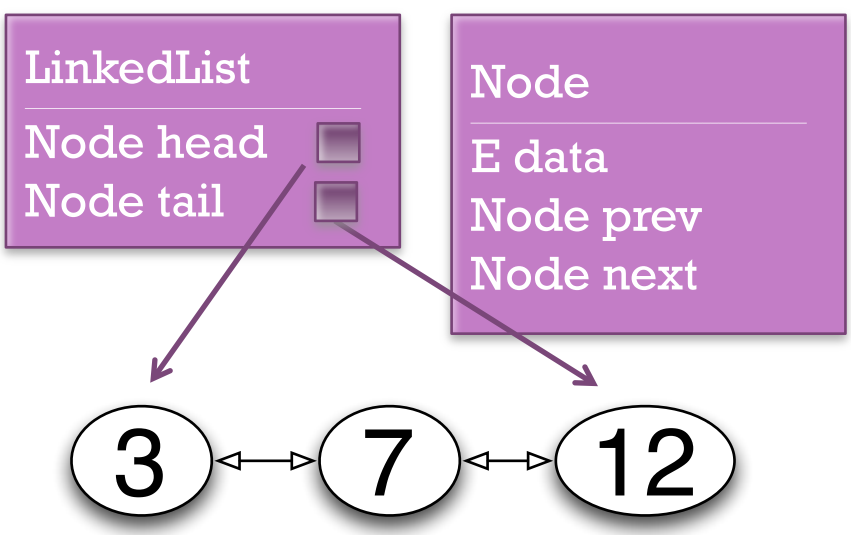
- **Full Binary Tree**
  All nodes have two children or 0 (leaf nodes)

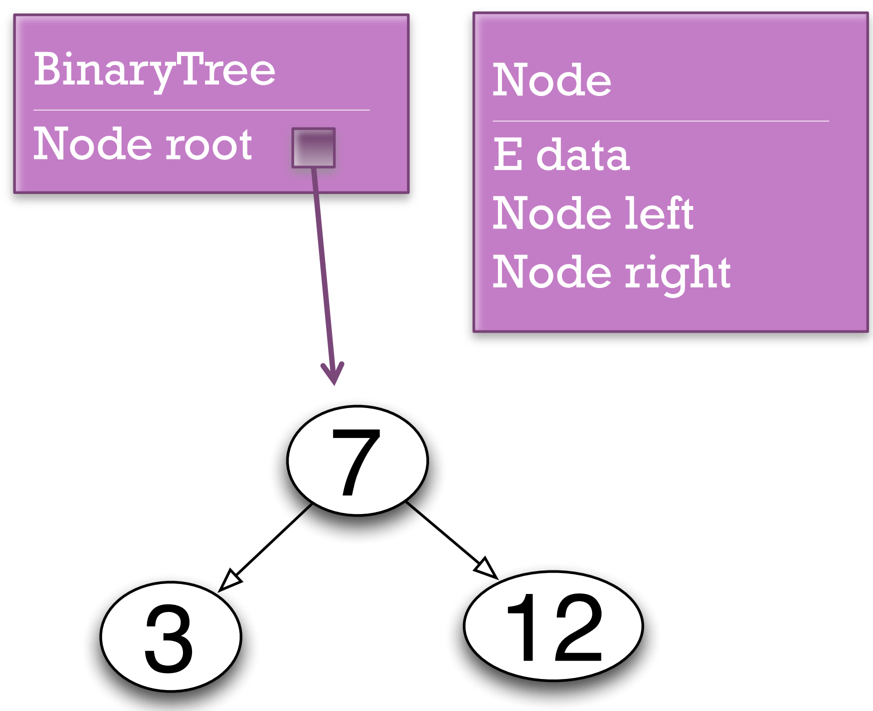
- **Perfect Binary Tree**
  Full Binary Tree of height n and $2^n - 1$ nodes.

- **Complete Binary Tree**
  Perfect through level n-1
  Extra leaf nodes at level n are all on left side of the tree.

+ Linked List vs. Binary Tree

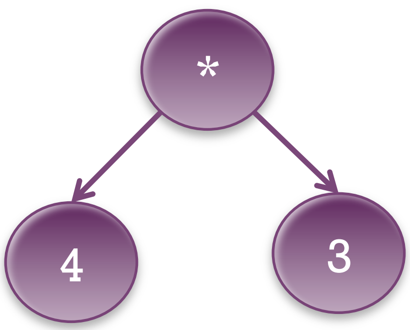
- Double linked list
  - A set of nodes
  - Each node has
    - Data
    - Edge to previous node
    - Edge to next node
  - head (and optional tail)
- Binary Tree
  - A set of nodes
  - Each node has
    - Data
    - Edge to left child
    - Edge to right child
  - A root node



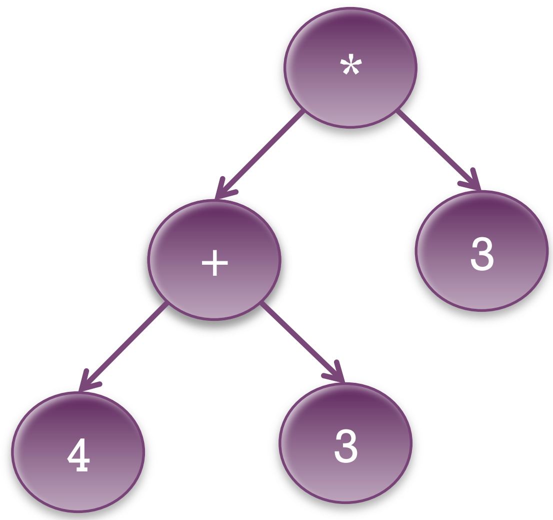


Expression Tree

- Operator at root (internal) nodes
- Operands at leaves (external) nodes
- 4 * 3



(4 + 3) * 3



- How would you draw:
- (x + y) * (a + b) / c

Binary Search Tree

- Nodes have
  - At most 2 children
  - One comparable value v
  - Any left subtree has values less than v
  - Any right subtree has values greater than v

Binary Tree Traversal

- How to make sure you "visit" each node only once.
- "Visiting" a node means that you operate on or use the value of the node.
- To demonstrate traversal, the value is often printed.
- Ordered Tree traversal:
  - Preorder: root, left, right
  - In-order: left, root, right
  - Post-order: left, right, root
- Example on board.

Traversal algorithms

- PreOrder(treeNode)
  - if TreeNode is empty
    - done
  - else
    - "visit" treeNode
    - PreOrder(treeNode.left)
    - PreOrder(treeNode.right)

- Simlarly for inOrder and postOrder
- Another example on board.

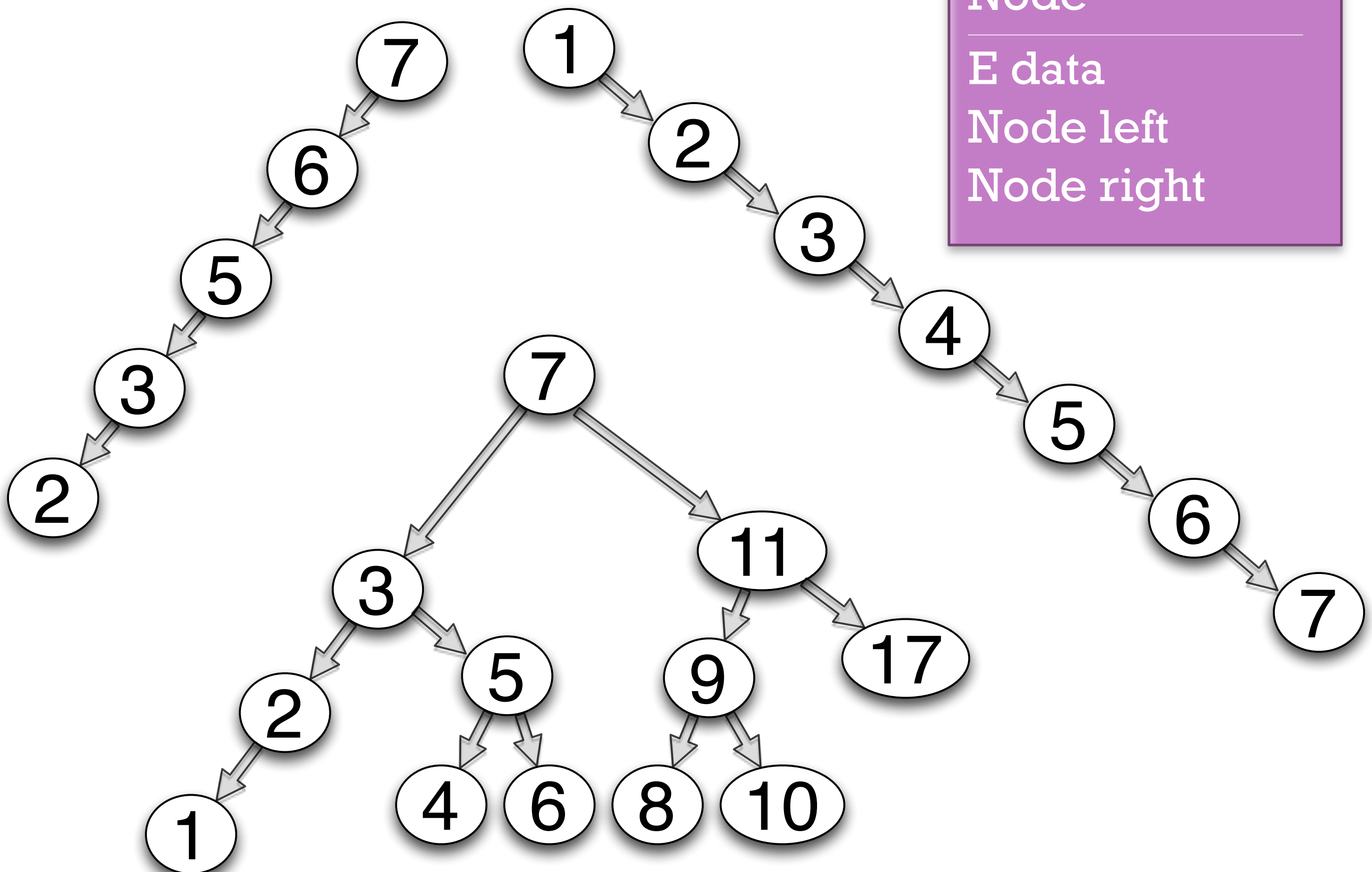
Other problems

- min
- max
- remove
- add

+ Min and Max?

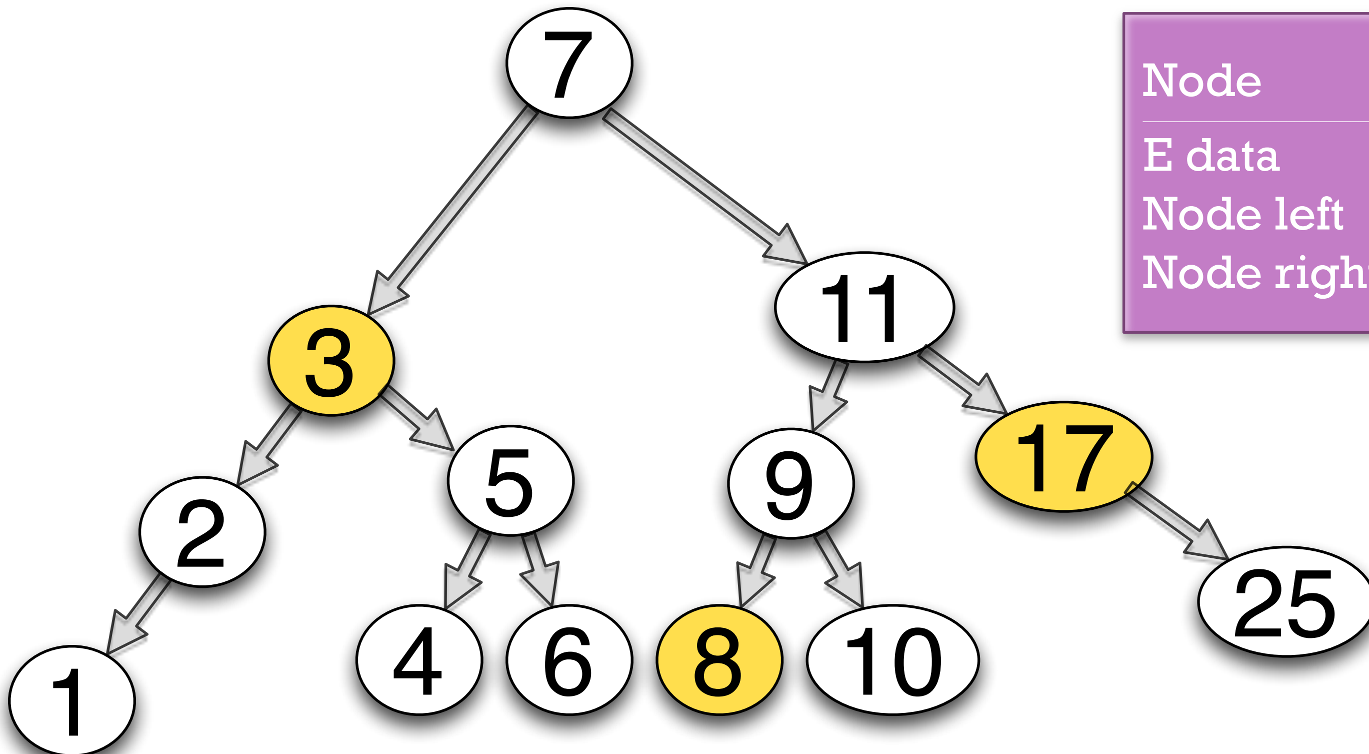
Node

E data
Node left
Node right



+ Removal

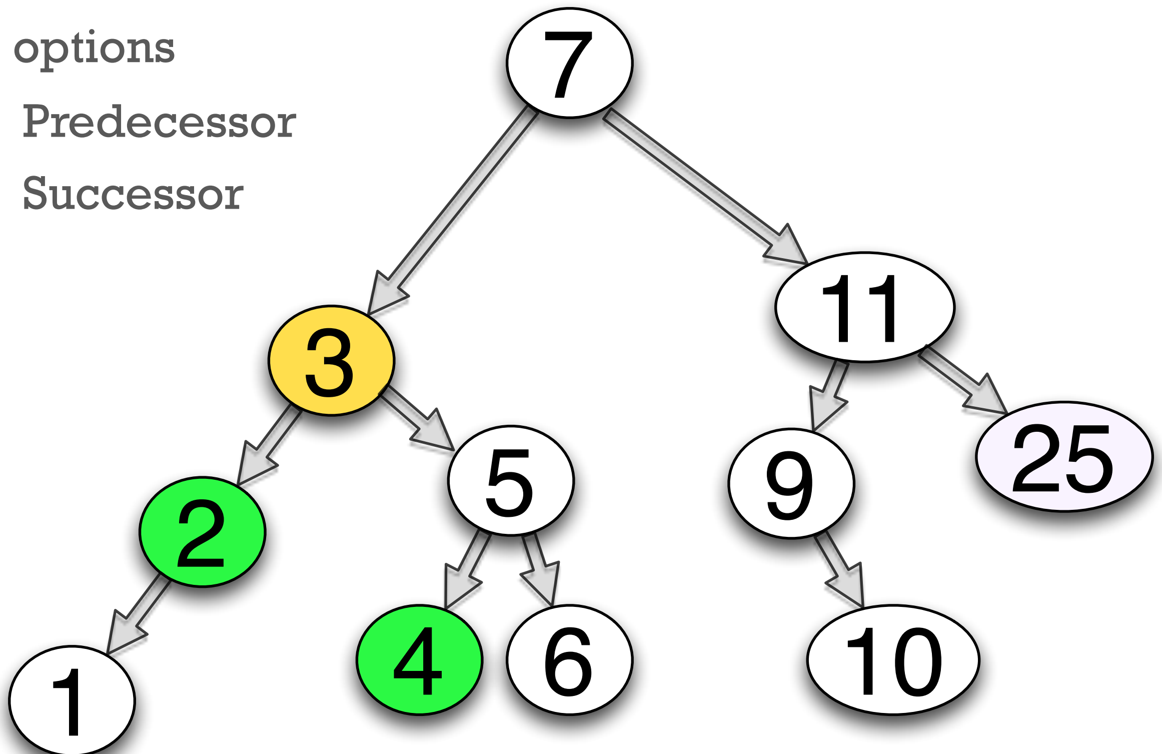
- Exercise: How do we remove 3, 8 and 17?
  - 3 cases



Node
E data Node left Node right

Removal

- Exercise: How do we delete?
  - 1 difficult case
    - 2 options
      - Predecessor
      - Successor



Node
E data Node left Node right