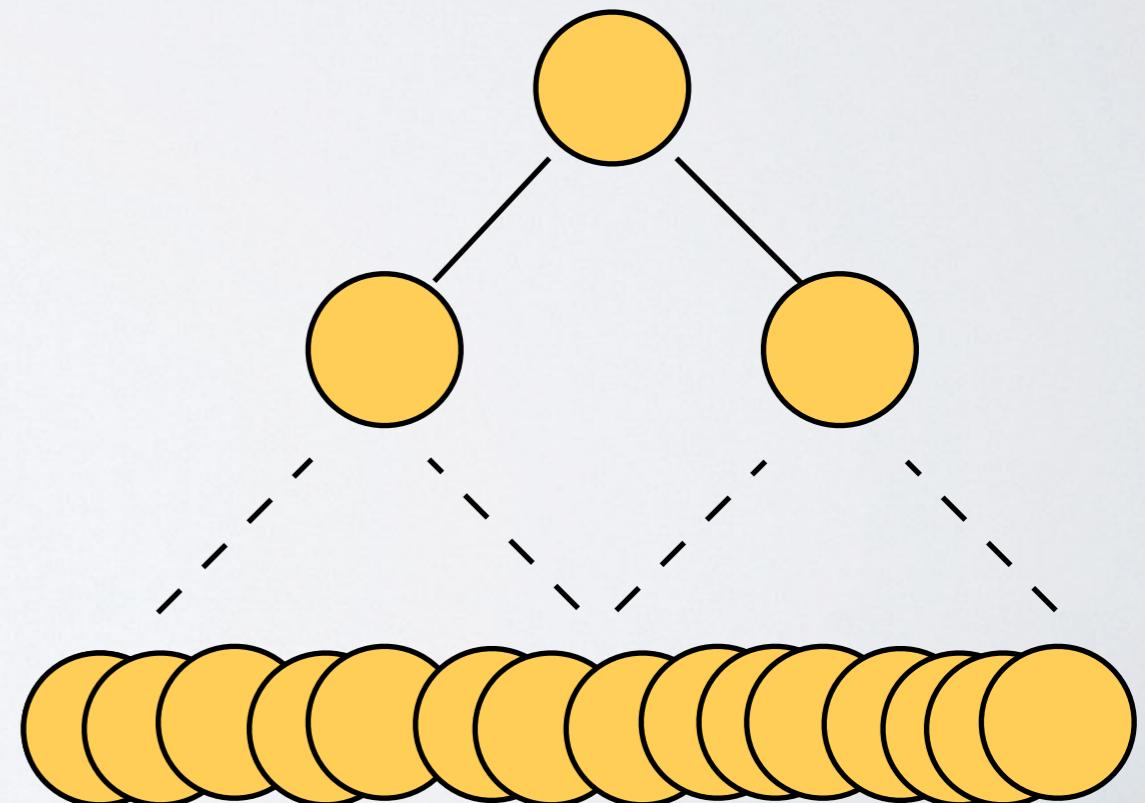


Binary Search Trees

CS16: Introduction to Data Structures & Algorithms
Summer 2021

Why trees, revisited, again

- ▶ Trees: natural representation of hierarchical data
 - ▶ Expression trees, directories, parse trees, etc.
- ▶ Also used for organizing data that **aren't** inherently hierarchical
- ▶ Why?
- ▶ Consider a perfect binary tree with N nodes
 - ▶ Height is $\log N$
- ▶ Want $O(\text{tree height})$ operations
- ▶ heap: insert, removeMin



Set ADT, revisited

- ▶ **add(object):**
 - ▶ adds object to set if not there
- ▶ **remove(object):**
 - ▶ removes object from set if there
- ▶ **boolean contains(object):**
 - ▶ checks if object is in set
- ▶ **int size():**
 - ▶ returns number objects in set
- ▶ **boolean isEmpty():**
 - ▶ returns TRUE if set is empty; FALSE otherwise
- ▶ **list enumerate():**
 - ▶ returns list of objects in set (in arbitrary order)



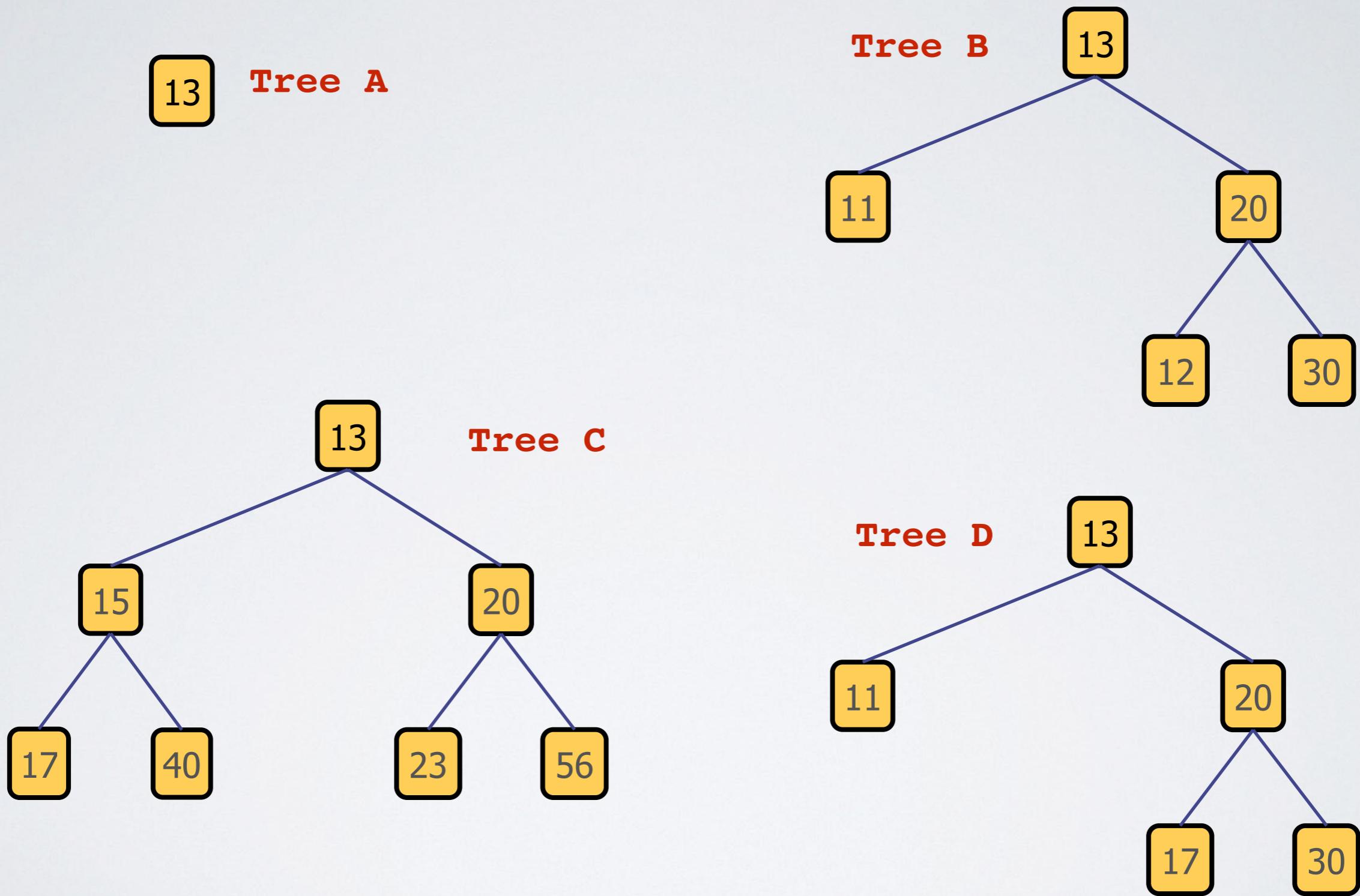
Implementing sets

- ▶ As we've seen: efficient implementation with hashing
- ▶ Why might we not want to use hashing?
 - ▶ Can't find a good hash function for data
 - ▶ Worried about worst case
 - ▶ Want to support range queries (more on which next time)
- ▶ Let's use trees instead!

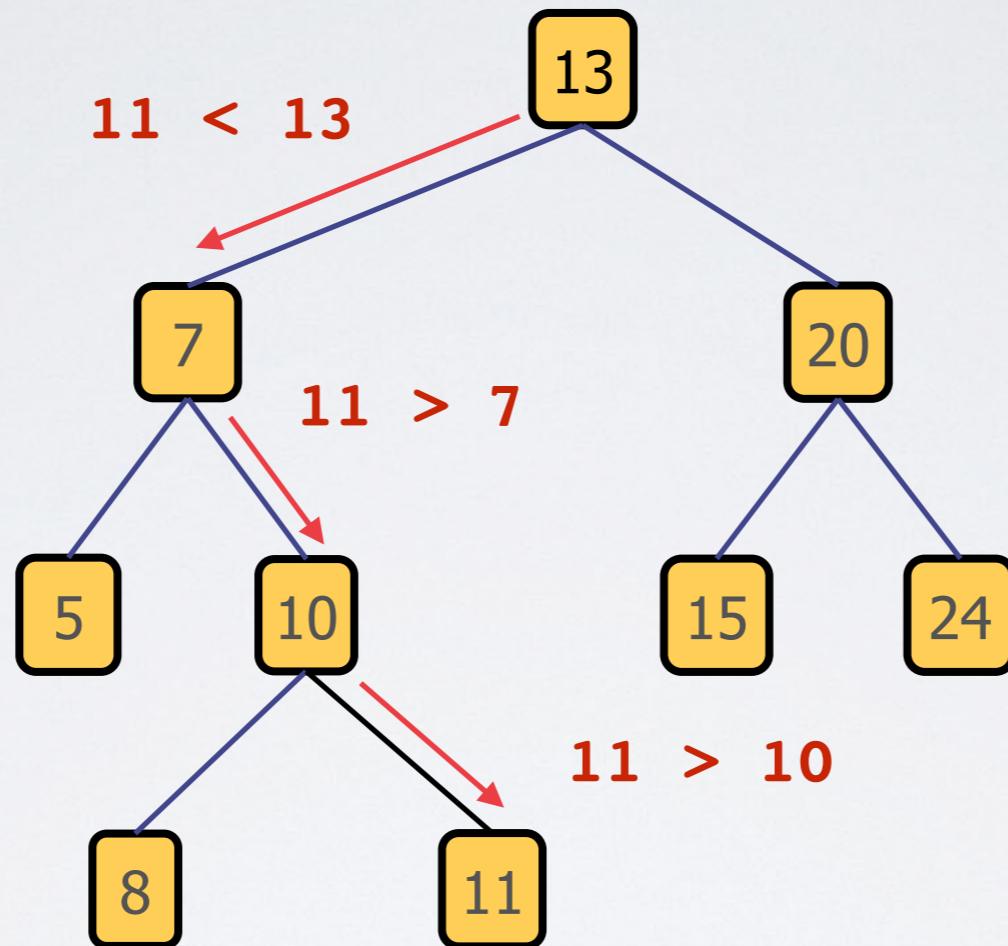
Binary Search Trees

- ▶ Binary trees with special property
 - ▶ For each node
 - ▶ left descendants have lower value than node
 - ▶ right descendants have higher value than node
- ▶ In-order traversal gives nodes in order

BSTs?

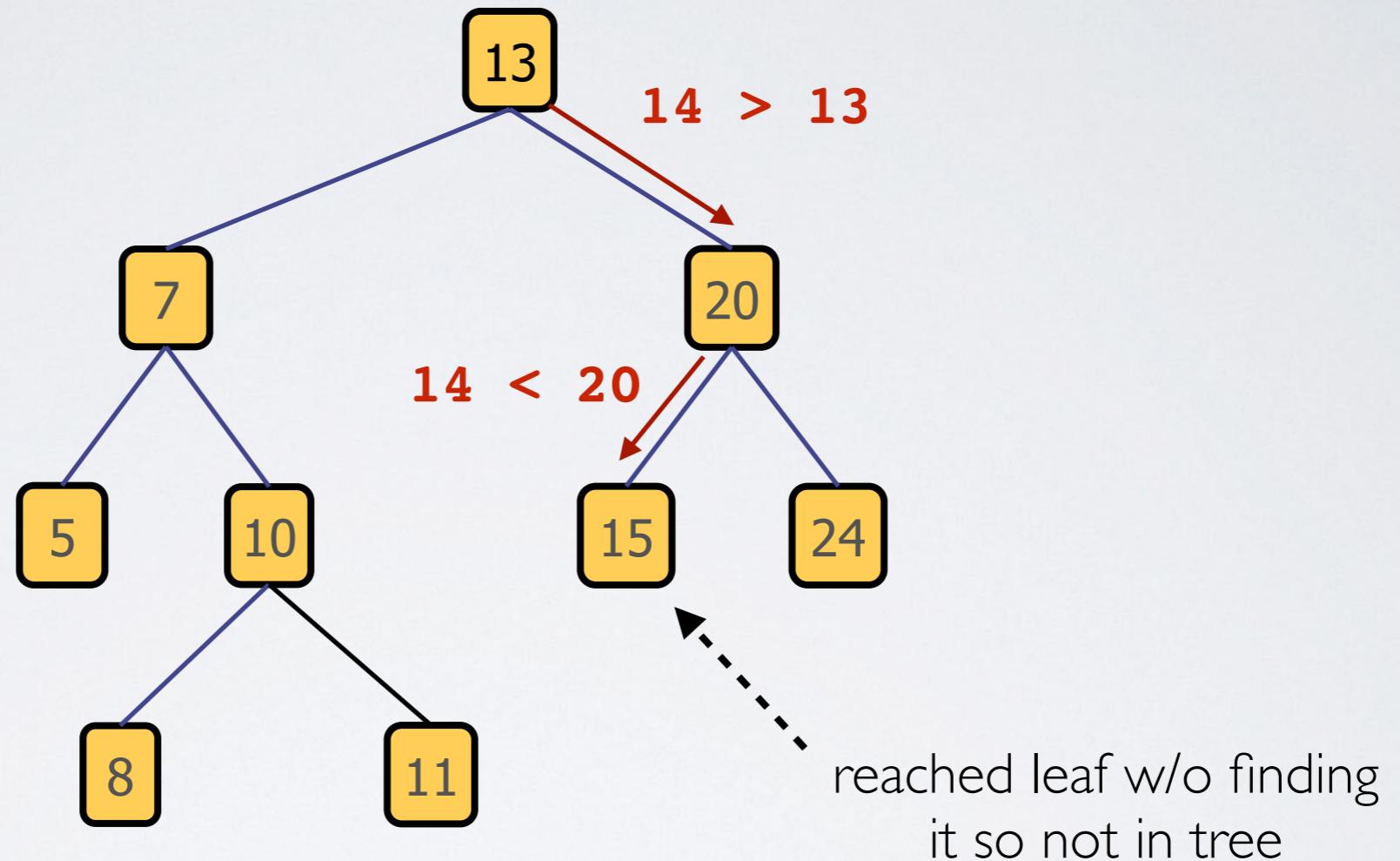


Searching a BST



- ▶ Find 11
- ▶ Each comparison tells us whether to go left or right

Searching a BST

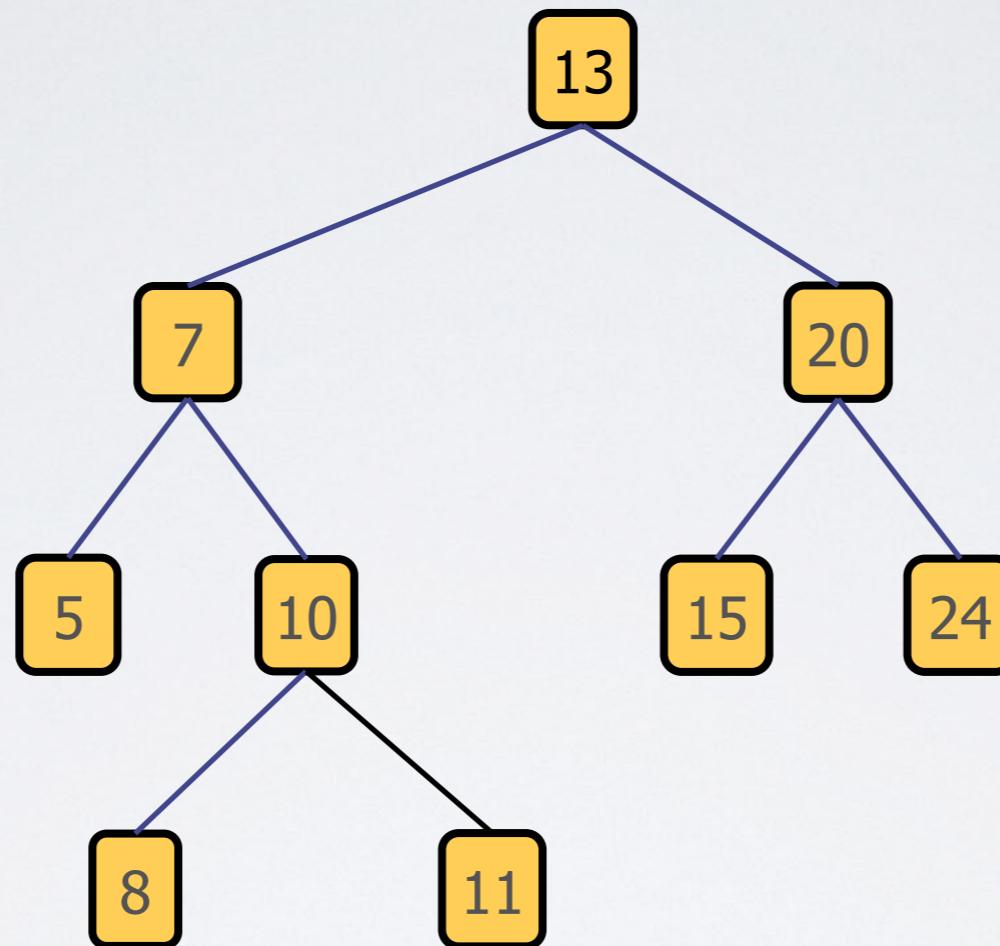


- ▶ What if item isn't in tree?
- ▶ Find 14

Binary Search Tree — Find()

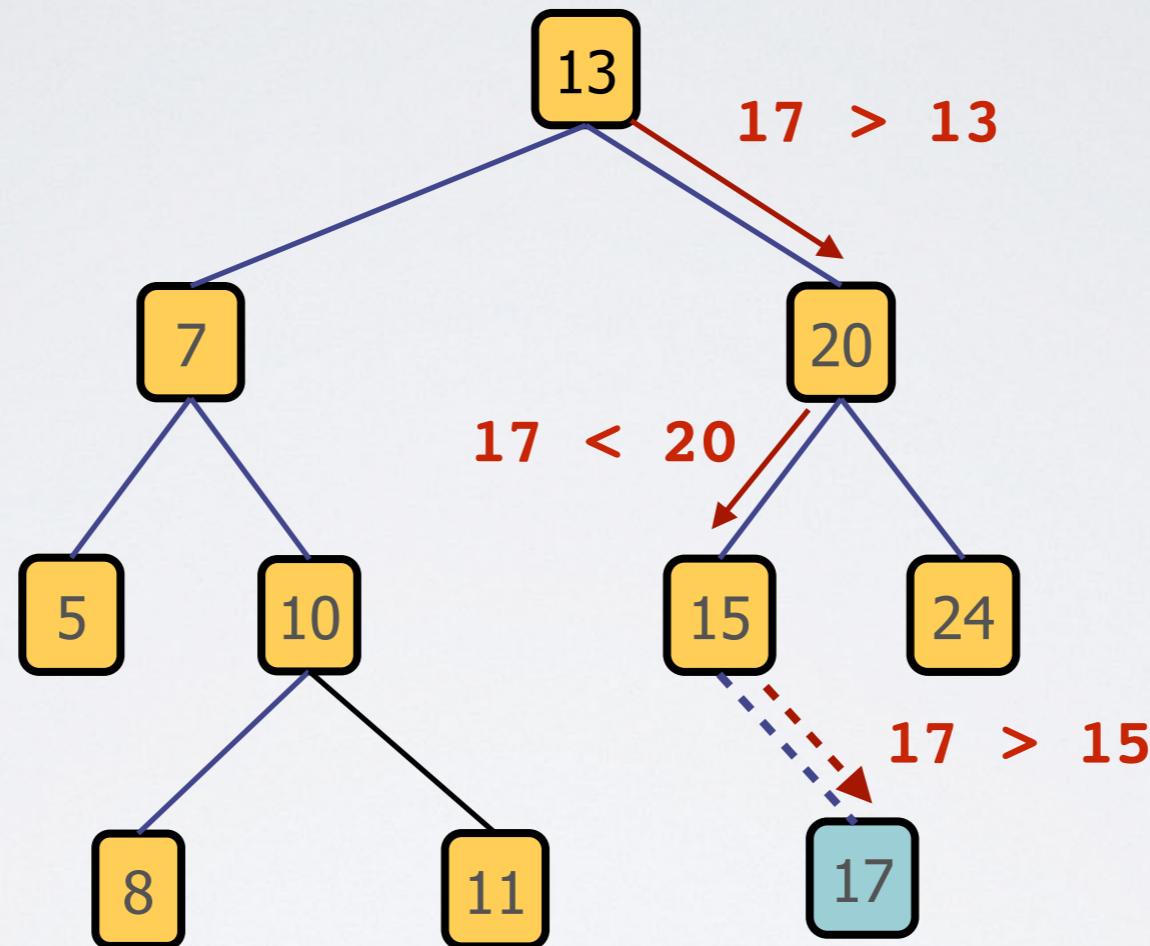
```
function find(node, toFind):  
    if node.data == toFind:  
        return node  
  
    else if toFind < node.data and node.left != null:  
        return find(node.left, toFind)  
  
    else if toFind > node.data and node.right != null:  
        return find(node.right, toFind)  
  
    return null
```

Inserting in a BST



- ▶ Want to insert 17, without modifying rest of tree
- ▶ So, have to add it as a child. Where does it go?

Inserting in a BST



- ▶ To insert, perform a search and add as new leaf
- ▶ Insert 17

Binary Search Tree — Insert()

```
function insert(node, toInsert):
    if node.data == toInsert: # data already in tree
        return

    if toInsert < node.data:
        if node.left == null: # add as left child
            node.addLeft(toInsert)
        else:
            insert(node.left, toInsert)
    else:
        if node.right == null: # add as right child
            node.addRight(toInsert)
        else:
            insert(node.right, toInsert)
```