Binary Trees

A binary tree is a tree that has a left branch and a right branch.
Idea: Fill the place of a missing left branch with an empty tree.
Idea 2: An instance of BTree always has exactly two branches.

```python
class BTree(Tree):
    empty = Tree(None)

def __init__(self, root, left=empty, right=empty):
    Tree.__init__(self, root, [left, right])

@property
def left(self):
    return self.branches[0]

@property
def right(self):
    return self.branches[1]
```

```python
t = BTree(3, BTree(1), BTree(7, BTree(5), BTree(9, BTree.empty, BTree(11))))
```

(Binary)

Binary Search Trees

Binary Search
A strategy for finding a value in a sorted list: check the middle and eliminate half.

20 in [1, 2, 4, 8, 16, 32, 64]

Finding 20:

```
[1, 2, 4, 8, 16, 32, 64]
```

4 in [1, 2, 4, 8, 16, 32]

Finding 4:

```
[1, 2, 4, 8, 16, 32]
```

For a sorted list of length \( n \), what Theta expression describes the time required? \( O(\log n) \).

Discussion Questions

- What’s the largest element in a binary search tree?
- def largest(t):
  if t.right is BTree.empty:
    return t.root
  elif t.is_leaf():
    return None
  else:
    return largest(t.right)

- What’s the second largest element in a binary search tree?
- def second(t):
  if t.right.is_leaf():
    return t.root
  elif t.right.is_leaf():
    return t.root
  else:
    return second(t.right)

(Binary)
Sets as Binary Search Trees

Membership in Binary Search Trees
contains traverses the tree
- If the element is not the root, it can only be in either the left or right branch
- By focusing on one branch, we reduce the set by the size of the other branch

```python
def contains(s, v):
    if s is BTree.empty:
        return False
    elif s.root == v:
        return True
    elif s.root < v:
        return contains(s.right, v)
    elif s.root > v:
        return contains(s.left, v)
```

Order of growth? \(\Theta(h)\) on average \(\Theta(\log n)\) on average for a balanced tree

Adjoining to a Tree Set

(Demo)