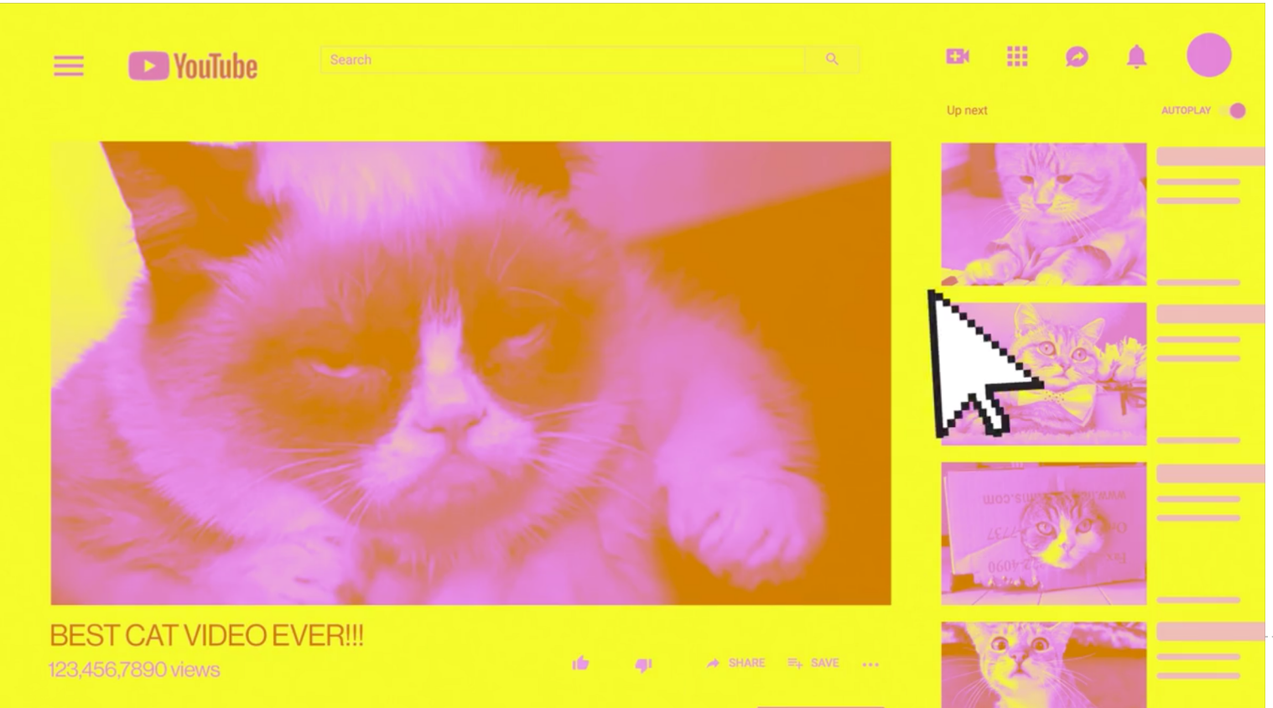
DeepMind is asking how Google helped turn the internet into an echo chamber

www.technologyreview.com/s/613083/deepmind-is-asking-how-google-helped-turn-the-internet-into-an-echo-chamber/



"One of the most common applications of machine learning today is in **recommendation algorithms**. Netflix and YouTube use them to push you new shows and videos; Google and Facebook use them to rank the content in your search results and news feed. While these algorithms offer a great deal of convenience, they have some undesirable side effects. You've probably heard of them before: filter bubbles and echo chambers.

Researchers found that the more accurately a recommendation engine pegs your interests, the faster it traps you in an information bubble.

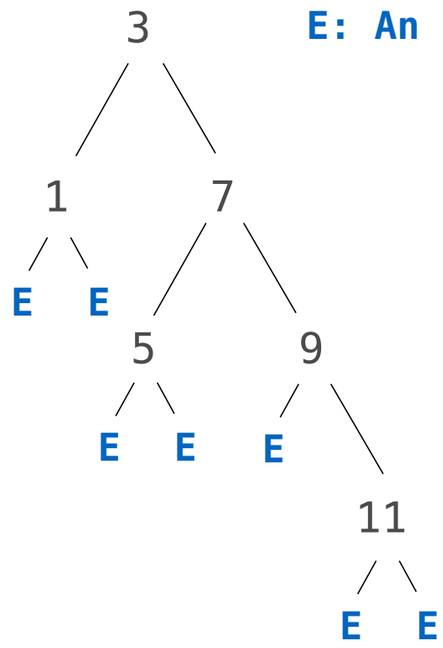
…**the best way to combat** filter bubbles or echo chambers **is to design the algorithms to be more exploratory, showing you things that are less certain to capture your interest**. Expanding the overall set of information from which the recommendations are drawn from can also help."

Binary Tree Class

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



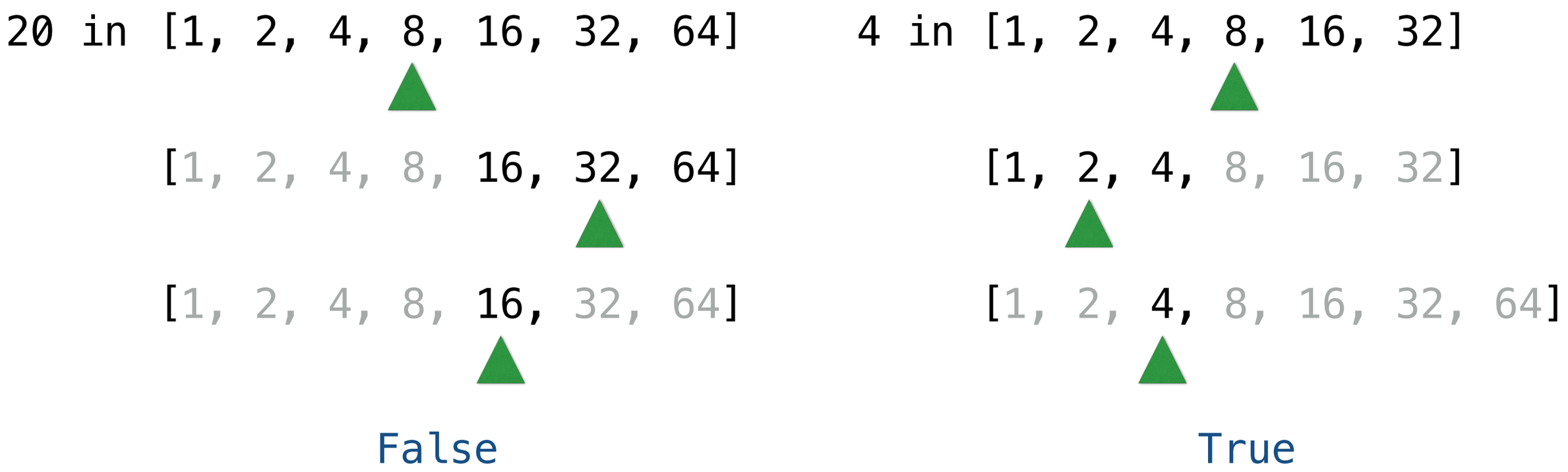
E: An empty tree

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

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

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

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


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

(Demo)

Binary Search

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

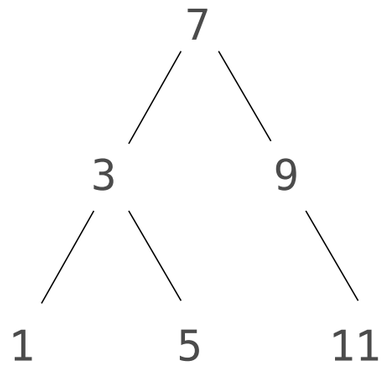


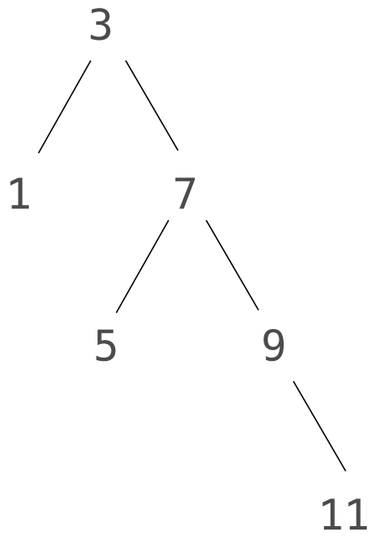
For a sorted list of length n, what Theta expression describes the time required? $\Theta(\log n)$

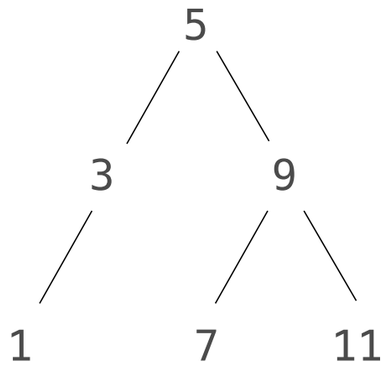
Binary Search Trees

A binary search tree is a binary tree where each node's label is:

- Larger than all node labels in its left branch and
- Smaller than all node labels in its right branch







(Demo)

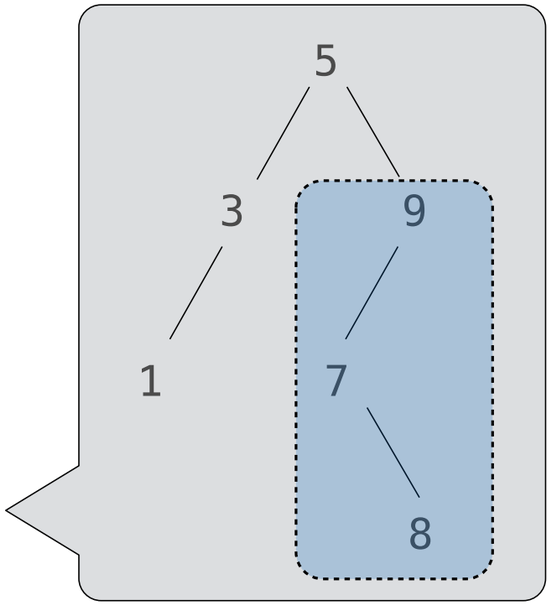
Discussion Questions

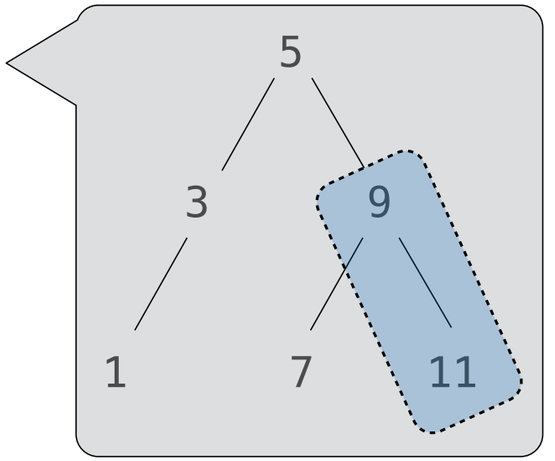
What's the largest element in a binary search tree?

```
def largest(t):
    if t.right is BTree.empty:
        return t.label
    else:
        return largest(t.right)
```

What's the second largest element in a binary search tree?

```
def second(t):
    if t.is_leaf():
        return None
    elif t.right is BTree.empty:
        return largest(t.left)
    elif t.right.is_leaf():
        return t.label
    else:
        return second(t.right)
```





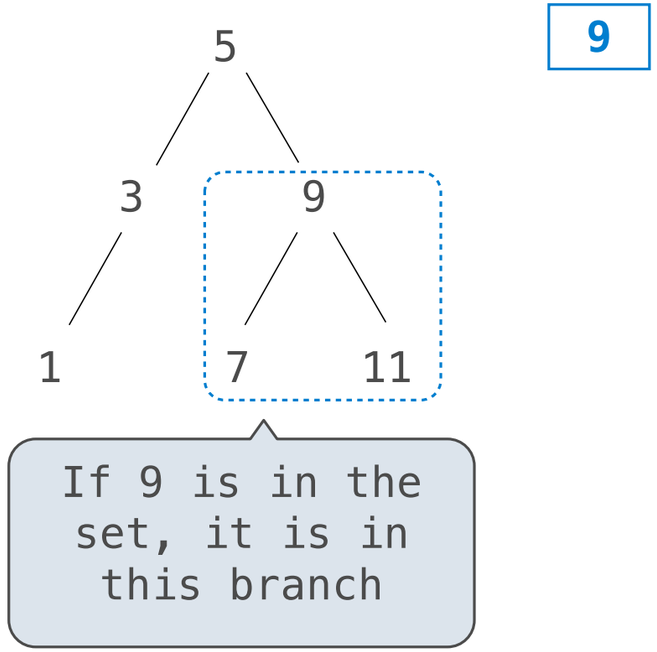
Sets as Binary Search Trees

Membership in Binary Search Trees

contains traverses the tree

- If the element is not at 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

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