при поддержке
{а}вижн

Лекция 21

Перевод и адаптация материалов Джона ДеНиро (John DeNero). Используется с разрешения автора.

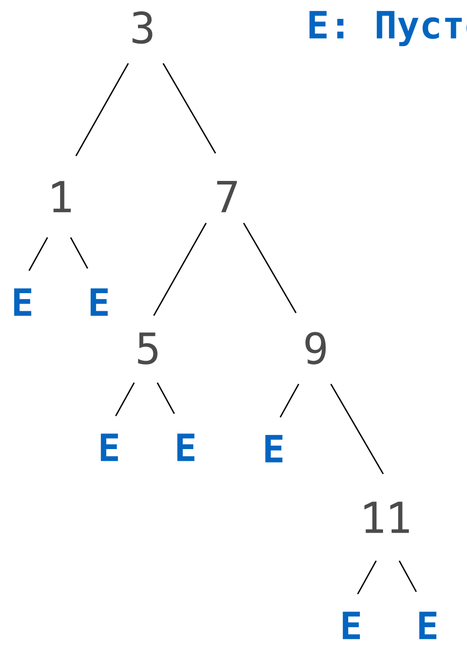
Двоичные деревья

Класс Binary Tree

Двоичное дерево – это дерево, у которого есть левая и правая ветвь.

Идея: Заполнять отсутствующие ветви пустыми деревьями.

Идея 2: Экземпляр BinaryTree всегда будет иметь в точности две ветви.



E: Пустое дерево

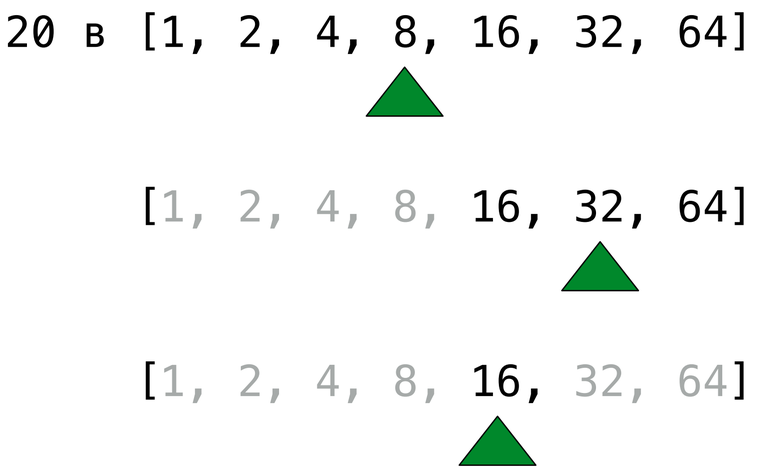
```
class BinaryTree(Tree):
    empty = Tree(None)

    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]
```

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

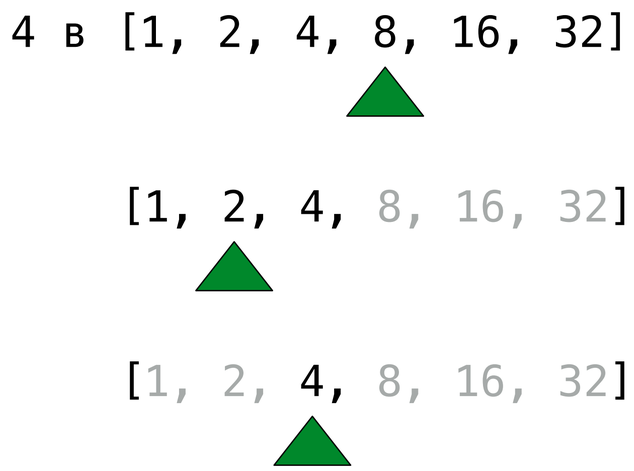
Двоичные деревья поиска

Двоичный поиск

Стратегия поиска значения в упорядоченном списке: проверка среднего значения и выбор первой или второй половины для дальнейшего поиска



False



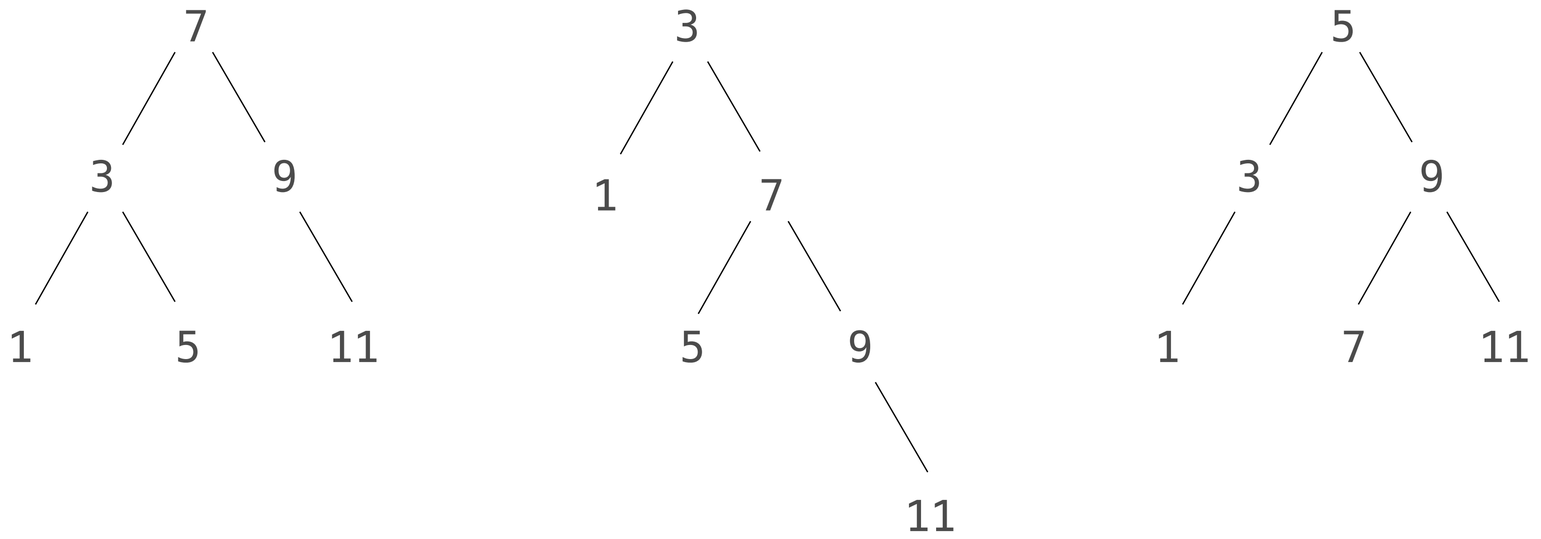
True

Для упорядоченного списка длины n, каков будет временной порядок роста? $\Theta(\log n)$

Двоичные деревья поиска

Двоичное дерево поиска — двоичное дерево, в котором значение узла:

- Больше всех узловых значений левой ветви
- Меньше всех значений правой ветви



(Пример)

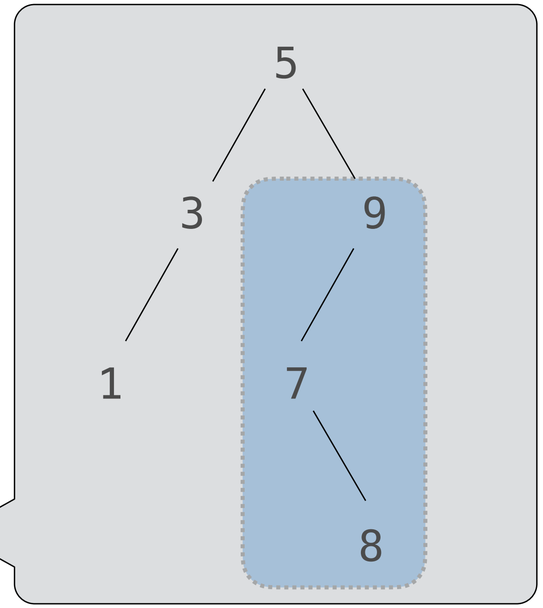
Вопрос

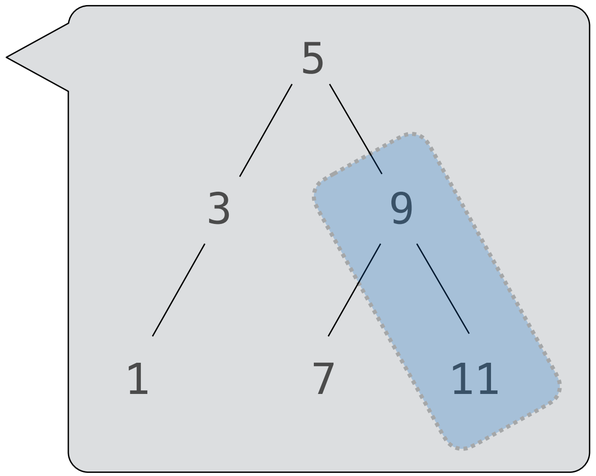
Каков наибольший элемент в BST?

```
def largest(t):
    if __t.right is BTree.empty__:
        return ____t.label____
    else:
        return _largest(t.right)_
```

Какой элемент в BST второй по величине?

```
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____
    return _second(t.right)_
```





Множества на двоичных деревьях поиска

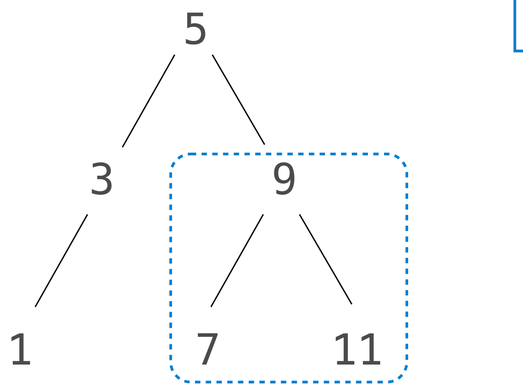
Поиск в двоичном дереве поиска

`contains` обходит дерево

- Если искомое значение v не узловое значение, то оно должно быть либо в правой, либо в левой ветви.
- Выбор одну ветви приводит к сокращению пространства поиска на размер второй ветви каждый рекурсивный вызов.

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

9



Если 9 входит в множество, надо искать в этой ветви

Порядок роста? $\Theta(h)$ в среднем $\Theta(\log n)$ в среднем, в сбалансированном дереве

Добавление элемента в множество (BST)