



数据结构与算法（Python版）

二叉查找树实现及算法分析（上）

陈斌 北京大学 gischen@pku.edu.cn

二叉搜索树的实现：BST.put方法

❖ put(key, val)方法：插入key构造BST

首先看BST是否为空，如果一个节点都没有，那么key成为根节点root

否则，就调用一个递归函数_put(key, val, root)来放置key

```
def put(self, key, val):  
    if self.root:  
        self._put(key, val, self.root)  
    else:  
        self.root = TreeNode(key, val)  
    self.size = self.size + 1
```

二叉搜索树的实现：_put辅助方法

❖ _put(key, val, currentNode)的流程

如果key比currentNode小，那么_put到左子树

- 但如果没有左子树，那么key就成为左子节点

如果key比currentNode大，那么_put到右子树

- 但如果没有右子树，那么key就成为右子节点

```
def _put(self, key, val, currentNode):  
    if key < currentNode.key:  
        if currentNode.hasLeftChild():  
            self._put(key, val, currentNode.leftChild)  
        else:  
            currentNode.leftChild = \  
                TreeNode(key, val, parent=currentNode)  
    else:  
        if currentNode.hasRightChild():  
            self._put(key, val, currentNode.rightChild)  
        else:  
            currentNode.rightChild = \  
                TreeNode(key, val, parent=currentNode)
```

二叉搜索树的实现：索引赋值

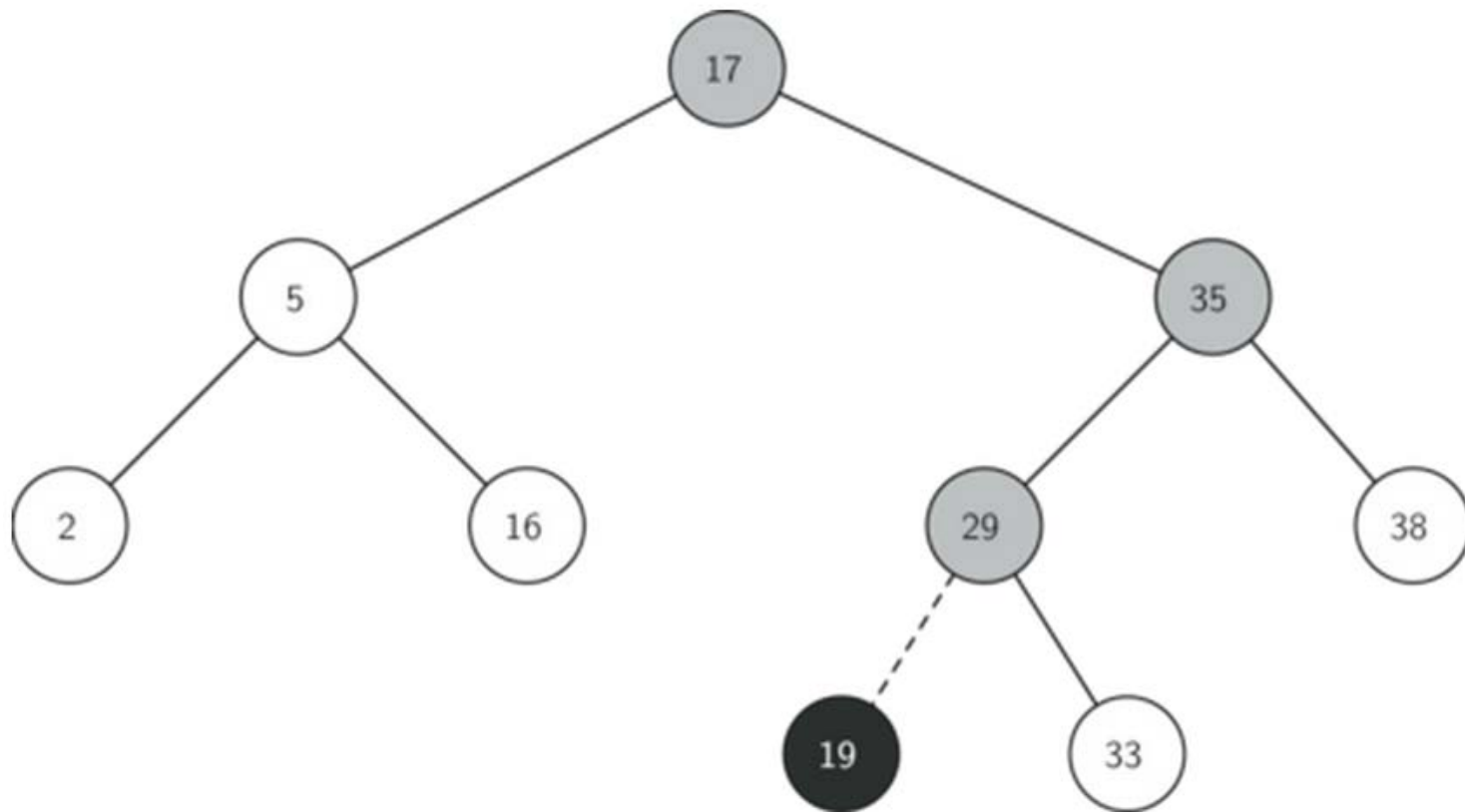
- ❖ 随手把__setitem__做了
- ❖ 特殊方法（前后双下划线）
可以myZipTree['PKU'] = 100871

```
def __setitem__(self, k, v):  
    self.put(k, v)
```

```
mytree = BinarySearchTree()  
mytree[3]="red"  
mytree[4]="blue"  
mytree[6]="yellow"  
mytree[2]="at"
```

二叉搜索树的实现：BST.put图示

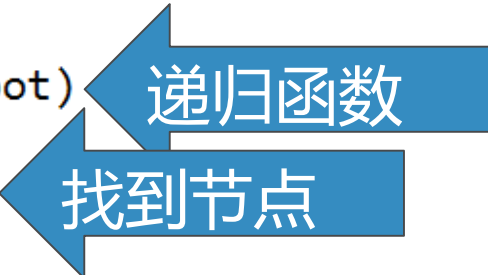
❖ 插入key=19, currentNode的变化过程
(灰色)：



二叉搜索树的实现：BST.get方法

❖ 在树中找到key所在的节点取到payload

```
def get(self, key):  
    if self.root:  
        res = self._get(key, self.root)  
        if res:  
            return res.payload  
        else:  
            return None  
    else:  
        return None  
  
def _get(self, key, currentNode):  
    if not currentNode:  
        return None  
    elif currentNode.key == key:  
        return currentNode  
    elif key < currentNode.key:  
        return self._get(key, currentNode.leftChild)  
    else:  
        return self._get(key, currentNode.rightChild)
```



二叉搜索树的实现：索引和归属判断

❖ `__getitem__` 特殊方法

实现 `val = myZipTree['PKU']`

❖ `__contains__` 特殊方法

实现 `'PKU' in myZipTree` 的归属判断运算符 `in`

```
def __getitem__(self, key):  
    return self.get(key)  
  
def __contains__(self, key):  
    if self._get(key, self.root):  
        return True  
    else:  
        return False
```

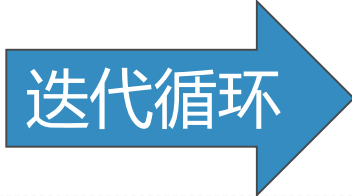
```
mytree[3] = "red"  
mytree[4] = "blue"  
mytree[6] = "yellow"  
mytree[2] = "at"  
  
print(3 in mytree)  
print(mytree[6])
```


二叉搜索树的实现：迭代器

- ❖ 我们可以用for循环枚举字典中的所有key
ADT Map也应该实现这样的迭代器功能
- ❖ 特殊方法 **`__iter__`** 可以用来实现for迭代
BST类中的 **`__iter__`** 方法直接调用了TreeNode
中的同名方法

```
mytree = BinarySearchTree()
mytree[3]="red"
mytree[4]="blue"
mytree[6]="yellow"
mytree[2]="at"

print(3 in mytree)
print(mytree[6])
del mytree[3]
print(mytree[2])
for key in mytree:
    print key, mytree[key]
```



迭代循环

二叉搜索树的实现：迭代器

❖ TreeNode类中的__iter__迭代器

迭代器函数中用了for迭代，实际上是递归函数

yield是对每次迭代的返回值

中序遍历的迭代

```
def __iter__(self):  
    if self:  
        if self.hasLeftChild():  
            for elem in self.leftChild:  
                yield elem  
        yield self.key  
        if self.hasRightChild():  
            for elem in self.rightChild:  
                yield elem
```

