九章算法

九章算法基础班

第七讲　集合、字典和分治法

课程版本：v2.0　　张三疯 老师

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏：http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

版权声明

九章课程不提供视频，也严禁录制视频的侵权行为

否则将追求法律责任和经济赔偿

请不要缺课

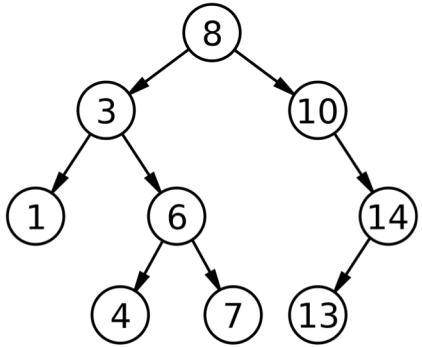
本节重点

- 集合（Set）和字典（Dictionary）
- 集合、字典的实现
- 分治算法（Divide and conquer）

课程回顾

宽度优先遍历

- 二叉树的宽度优先遍历的实现
  - 使用队列（Queue）作为主要的数据结构
  - 代码演示



宽度优先遍历

- 二叉树的宽度优先遍历
  - 分层遍历
  - 多一个循环
  - 代码演示

- 层数记录了根节点到当前节点的路径长度

宽度优先遍历

- 二叉树的宽度优先遍历
  - 时间复杂度：O(n)
  - 空间复杂度：由节点最多的层的节点数决定，O(n)

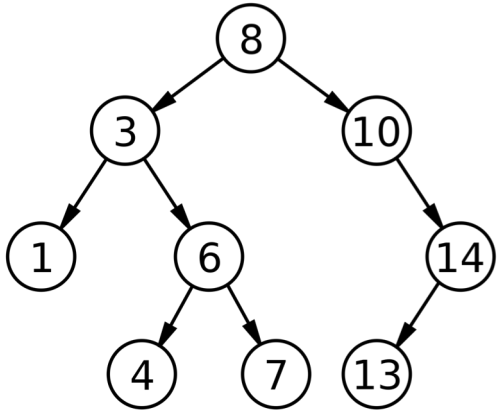
BST

- 什么是BST（Binary Search Tree）
  - 满足以下性质的Binary tree
  - 对于每个节点，他的左子树的所有节点都比它小
  - 对于每个节点，他的右子树的所有节点都比它大

- 以上是严格的说法，不允许BST有重复的节点，实际算法中可以允许重复

BST

- BST的特性有哪些应用？
  - 对BST进行中序遍历，得到的是一个非降序的序列
  - 插入操作（普通的Binary tree没有插入操作的概念）
  - 高效的查找



平衡BST的应用

- 高效插入（insert）和查找（find）元素

- 在实际系统中广泛应用
  - 数据库
  - 搜索引擎

集合（Set）和字典（Dictionary）

集合（Set）

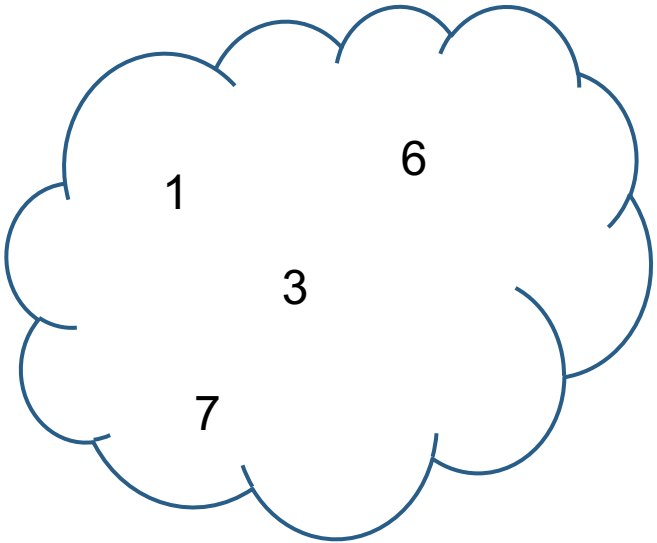
- Python的基本数据结构之一
  - 集合中存储非重复的无序数据
  - set中的元素不一定是同一类型，非常灵活

```
set_1 = set([12, 15.6, True, 'hello'])
set_2 = {12, 15.6, True, 'hello'}
set_3 = set('hello')
```

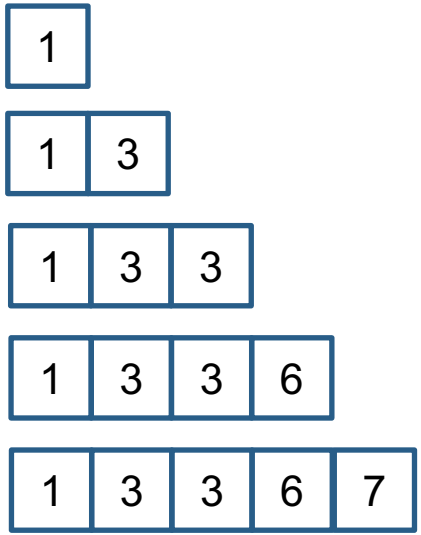
Bag

集合（Set）

Set



List



集合（Set）

- Set的常见操作
  - 增（Create）：add, update
  - 查（Read）：迭代（iteration）, in
  - 删（Delete）：remove, clear, discard, pop
  - 其他：len

集合（Set）

- Set的集合间操作：生成新的集合
  - 并集：union, |
  - 交集：intersection， &
  - 差集： difference， -
  - 对称差： symmetric_difference，^

集合（Set）

九章算法

- Set的插入和查找操作效率很高
  - 非常适合用来记录某元素是否出现过

- 练习一：面试真题
  - Remove Duplicate Numbers in Array
  - https://lintcode.com/problem/remove-duplicate-numbers-in-array/
  - https://www.jiuzhang.com/solution/remove-duplicate-numbers-in-array/

字典（Dictionary）

九章算法

- Python的基本数据结构之一
  - 字典中存储key非重复的无序的key-value pairs
  - dict可能是最灵活的内置数据结构
  - 别名：index，map

```
dict_1 = {}
dict_2 = {'spam': 2, 'eggs': 3, 'food': {'ham': 1, 'ice': 2}}
dict_3 = dict(zip(['spam', 'eggs'], [2, 3]))
```

字典（Dictionary）

九章算法

- Dict的常见操作
  - 增（Create）：索引赋值，update
  - 查（Read）：索引，迭代（iteration），in, get, keys, values, items
  - 改（Update）：索引赋值
  - 删（Delete）：pop, del
  - 其他：len

字典（Dictionary）

- dict的插入和查找操作效率很高
  - 可以用来记录某元素是否出现过
  - 同时还可以附带一个属性值（如：元素的位置）

字典（Dictionary）

九章算法

- 练习二
  - Two Sum
  - https://lintcode.com/problem/two-sum/
  - https://www.jiuzhang.com/solution/two-sum/

集合和字典

- set的元素和dict的key必须是可以hash的
  - list不可以，tuple可以
  - set不可以， frozenset可以
  - dict不可以

集合和字典的实现

- 主要关注插入和查找操作
  - List实现

    insert: O(n) find: O(n)
  - 平衡的BST实现

    insert: O(logn) find: O(logn)
  - Hash table实现

    insert: O(1) find: O(1)

集合和字典的实现

- BST vs. Hash table
  - Hash table的插入，查找和删除操作都是O(1)时间，但空间消耗大
  - BST支持有序的数据，排序和范围查找性能优秀（对于平衡的BST是O(logn)），空间消耗小

集合和字典的实现

- Python中的set和dict使用hash table实现
  - 要求set的元素和dict的key是可hash的

	Balanced BST	Hash table
C++	set / map	unordered_set / unordered_map
Java	TreeSet / TreeMap	HashSet / HashMap
Python	? ? ?	set / dict

分治算法

分治法

- 分治法（divide and conquer）
  - 将一个大问题分解成多个独立的小问题：分
  - 分别解决每个小问题（小问题和大问题是同一类问题，可以用递归）
  - 将小问题的解合并，从而得到大问题的解：合

分治法

- 练习三：面试真题
  - Identical Binary Tree
  - https://www.lintcode.com/problem/identical-binary-tree/
  - https://www.jiuzhang.com/solution/identical-binary-tree/

分治法

- Identical Binary Tree
  - 判断两棵树根的值是否相同
  - 判断左子树是否一致
  - 判断右子树是否一致

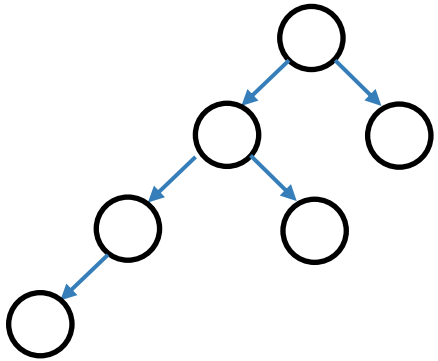
分治法

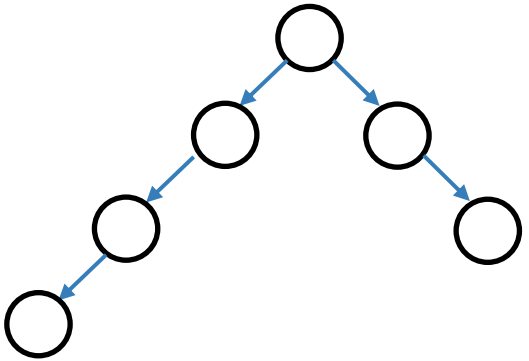
九章算法

- 练习四：面试真题
  - Balanced Binary Tree
  - https://lintcode.com/problem/balanced-binary-tree/
  - https://www.jiuzhang.com/solution/balanced-binary-tree/

分治法

- Balanced Binary Tree
  - 判断左子树是否平衡
  - 判断右子树是否平衡
  - 左右子树高度差不超过1





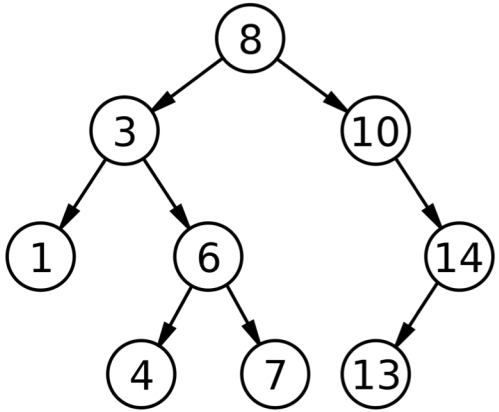
分治法

九章算法

- 练习五：面试真题
  - Validate Binary Search Tree
  - https://lintcode.com/problem/validate-binary-search-tree/
  - https://www.jiuzhang.com/solution/validate-binary-search-tree/

分治法

- Validate Binary Search Tree
  - 判断左子树是否是BST
  - 判断右子树是否是BST
  - 左子树的最大值 ≤ 根节点的值 ≤ 右子树的最小值



总结

- 集合（Set）和字典（Dictionary）
  - 常见操作
  - 不同实现方式的优缺点

- 分治法（Divide and conquer）
  - 先分再合

调查问卷

- https://www.jiuzhang.com/course/13/questionnaire/

Q&A

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

谢谢大家