<http://www.jianshu.com/p/42f81846c0fb>

# 概念

## 算法

算法是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。对于同一个问题的解决，可能会存在着不同的算法，为了衡量一个算法的优劣，提出了空间复杂度与时间复杂度这两个概念。

## 时间复杂度

## 空间复杂度

# 数据结构

## 线性结构

### 数组

### 链表

### 队列

### 栈

## 非线性结构

### 树

#### 二叉查找树

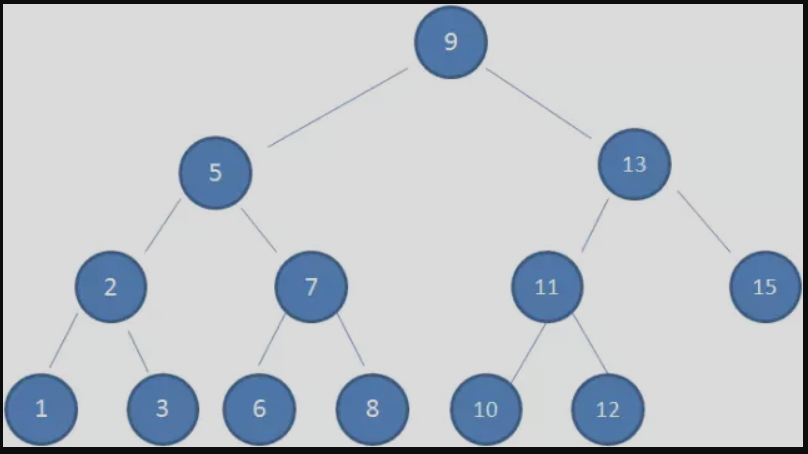
<http://mp.weixin.qq.com/s?__biz=MzIxNTIwOTgxNw==&mid=2650612655&idx=1&sn=09bd44c8bf9978c8beedcdf5a07d1047&chksm=8f924199b8e5c88f45a0ffc8571c820d830c4a332a2c16f162d48295e70ea796ebabc4773fdf&mpshare=1&scene=23&srcid=1125u1TfchKscPuptKpdLlMC#rd>

**特性**

**1. 左子树上所有结点的值均小于或等于它的根结点的值。**

**2. 右子树上所有结点的值均大于或等于它的根结点的值。**

**3. 左、右子树也分别为二叉排序树。**



问：找出值为10的节点

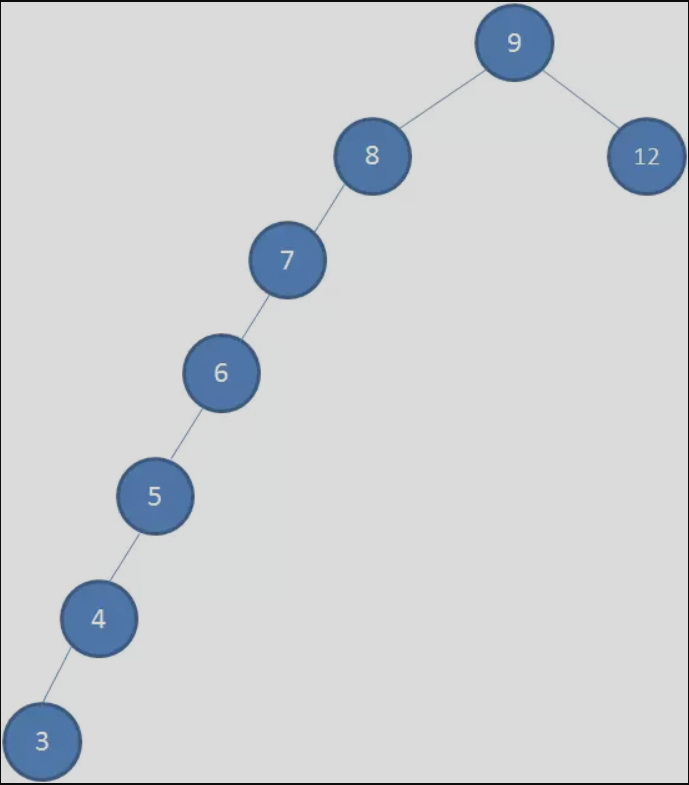
先看9,9小于10，根据二叉查找树的特性，10应该在节点9的右节点子树上。

依次查看9-13-11->9，可找到9所在节点。

这种方式正是二分查找的思想，查找所需的最大次数等于二叉树的高度。插入节点时也是基于该特性。

**缺陷**

依次插入如下五个节点：7,6,5,4,3。依照二叉查找树的特性，结果会变成什么样呢？



虽然符合二叉查找数的特性，但查找效率大打折扣，几乎变成线性。

为了**解决二叉查找树多次插入新节点导致的不平衡**，红黑树应运而生。

#### 红黑树

自平衡的二叉树，符合二叉查找树的基本特性，此外还具有如下特性：

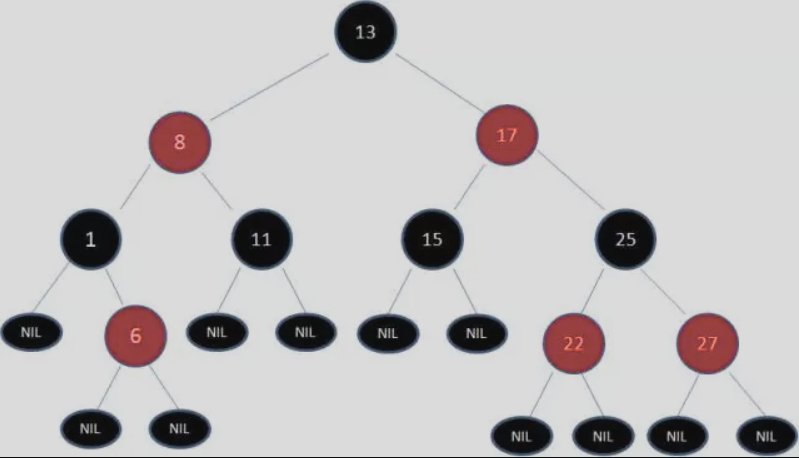
**1. 节点是红色或黑色。**

**2. 根节点是黑色。**

**3. 每个叶子节点都是黑色的空节点（NIL节点）。**

**4. 每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)**

**5. 从任一节点到其每个叶子的所有路径都包含相同数目的黑色节点。**

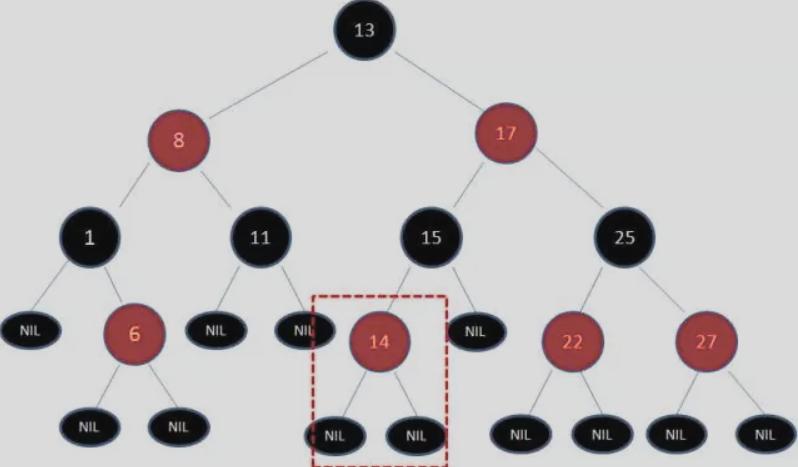


查找值为10的节点：13-8-11-NULL

红黑树从根到叶子的最长路径不会超过最短路径的2倍。

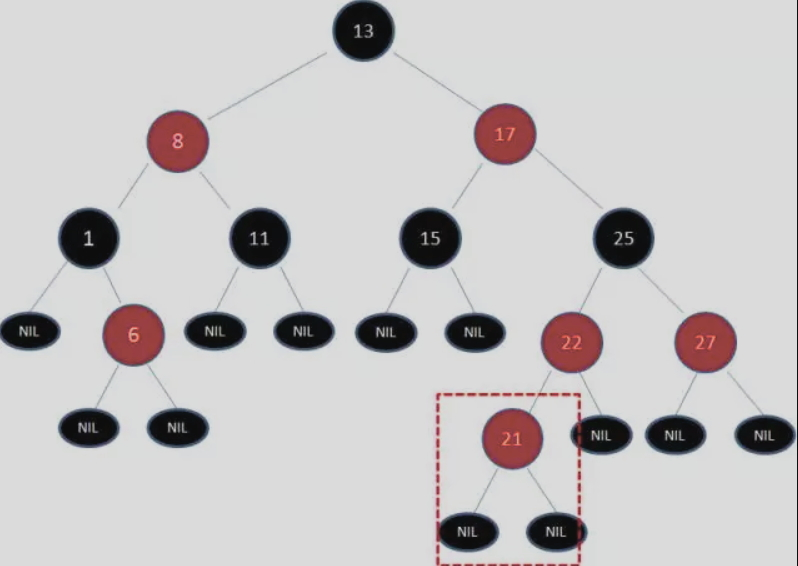
当插入或删除节点时，红黑树的规则有可能被打破，此时需要做出调整，来继续维持红黑树的规则。

1. 向原红黑树插入值为14的新节点：



由于父节点15是黑色节点，因此这种情况并不会破坏红黑树的规则，无需做任何调整。

1. 向原红黑树插入值为21的新节点：



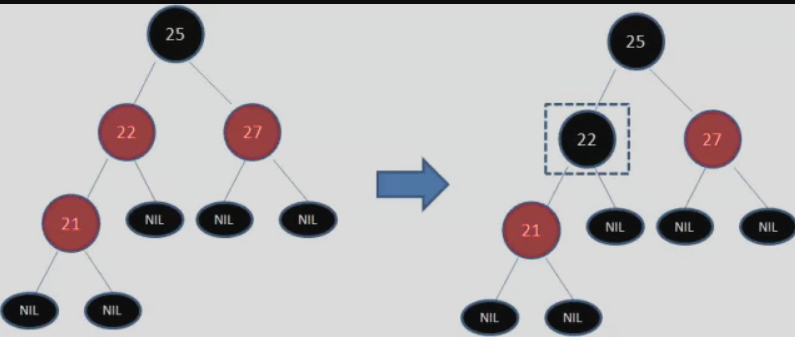
由于父节点22是红色节点，因此这种情况打破了红黑树的规则4（每个红色节点的两个子节点都是黑色），必须进行调整，使之重新符合红黑树的规则。

调整有两种方法：**变色**和**旋转**（旋转又分左旋转和右旋转）。

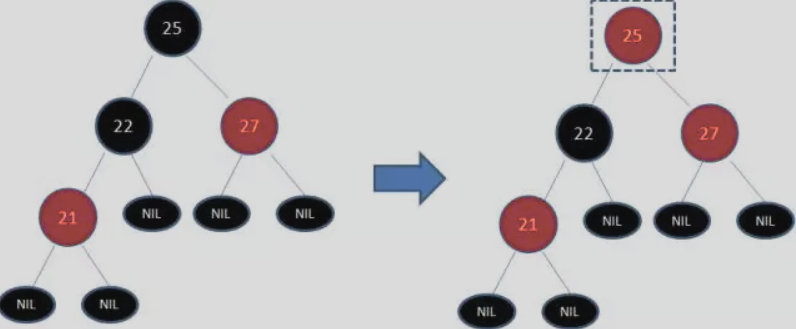
* 变色

为了重新符合红黑树的规则，尝试把红色节点变为黑色，或者把黑色节点变为红色。

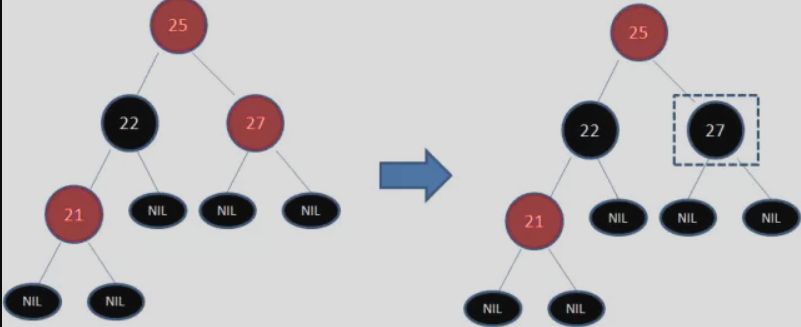
下图所表示的是红黑树的一部分，需要注意节点25并非根节点。因为节点21和节点22连续出现了红色，不符合规则4，所以把节点22从红色变成黑色：



但这样并不算完，因为凭空多出的黑色节点打破了规则5，所以发生连锁反应，需要继续把节点25从黑色变成红色：



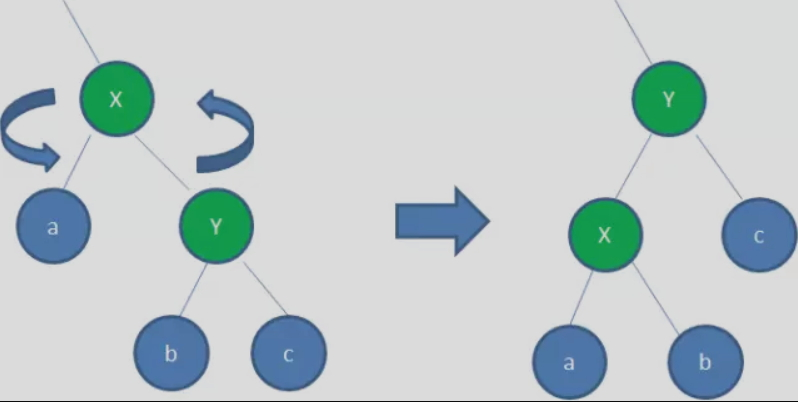
此时仍然没有结束，因为节点25和节点27又形成了两个连续的红色节点，需要继续把节点27从红色变成黑色：



* 旋转

左旋转

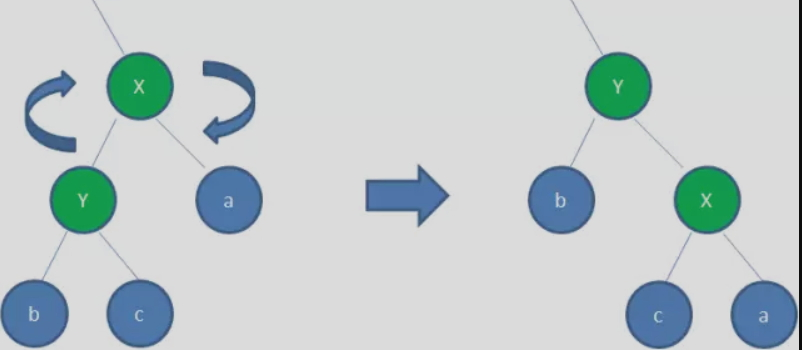
逆时针旋转红黑树的两个节点，使得父节点被自己的右孩子取代，而自己成为自己的左孩子。说起来很怪异，大家看下图：



图中，身为右孩子的Y取代了X的位置，而X变成了自己的左孩子。此为左旋转。

右旋转

顺时针旋转红黑树的两个节点，使得父节点被自己的左孩子取代，而自己成为自己的右孩子。大家看下图：



图中，身为左孩子的Y取代了X的位置，而X变成了自己的右孩子。此为右旋转。

实际应用

JDK集合类，TreeMap和TreeSet底层用的是红黑树，java8中HashMap也用到了红黑树。

### 图

### 表

## 哈希结构

# 算法

## 排序

<http://www.jianshu.com/p/42f81846c0fb>

### 冒泡排序

### 简单选择排序

### 直接插入排序

### 归并排序

### 希尔（Shell）排序

### 快速排序

### 堆排序



## 查找

### 顺序查找

### 二分查找

## Dijkstra （最短路径算法）

## 递归

## 分治算法

## 动态规划

## 贪心算法

## 回溯算法

## 匹配算法

## 正则表达式和字符串匹配

# 习题

<http://www.codeceo.com/article/15-algorithms-question.html>

<http://dongxicheng.org/structure/structure-algorithm-summary/>