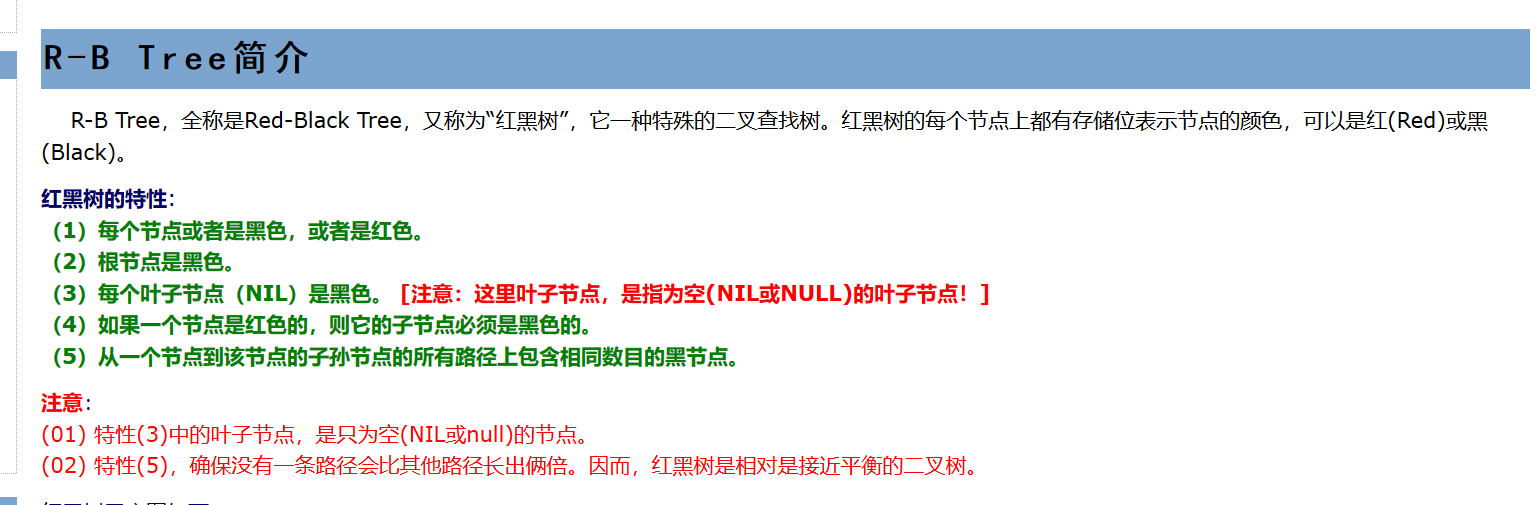
## 红黑树的定义

红黑树是满足一些性质的二叉搜索树 ,二叉搜索树也称作AVL树



## 红黑树的时间复杂度以及特性

,一颗含有n个节点的红黑树,红黑树的时间复杂度是log2 n,树高最多为 2 (log2 (n+1) )



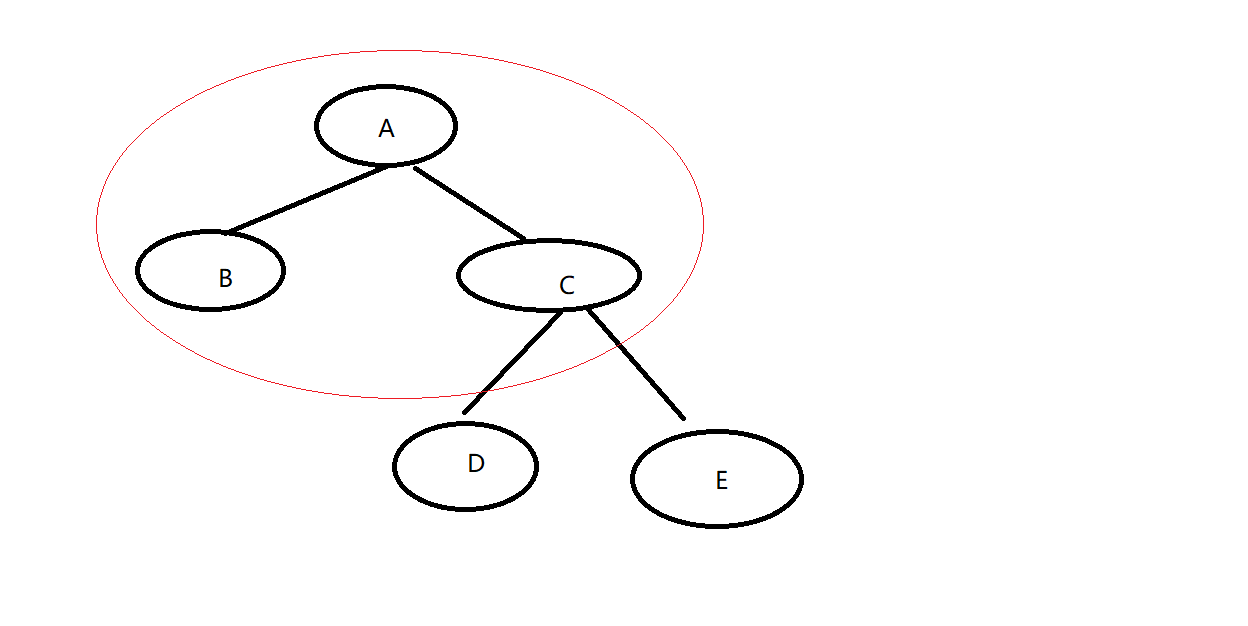
## 红黑树的优点

传统的二叉搜索树中,由于树可能出现只有左子树情况,大大的浪费了空间以及增加了时间复杂度.对于树的来说, 高度尽量矮胖对于时间复杂度最好

## 红黑树中的左旋和右旋的说明

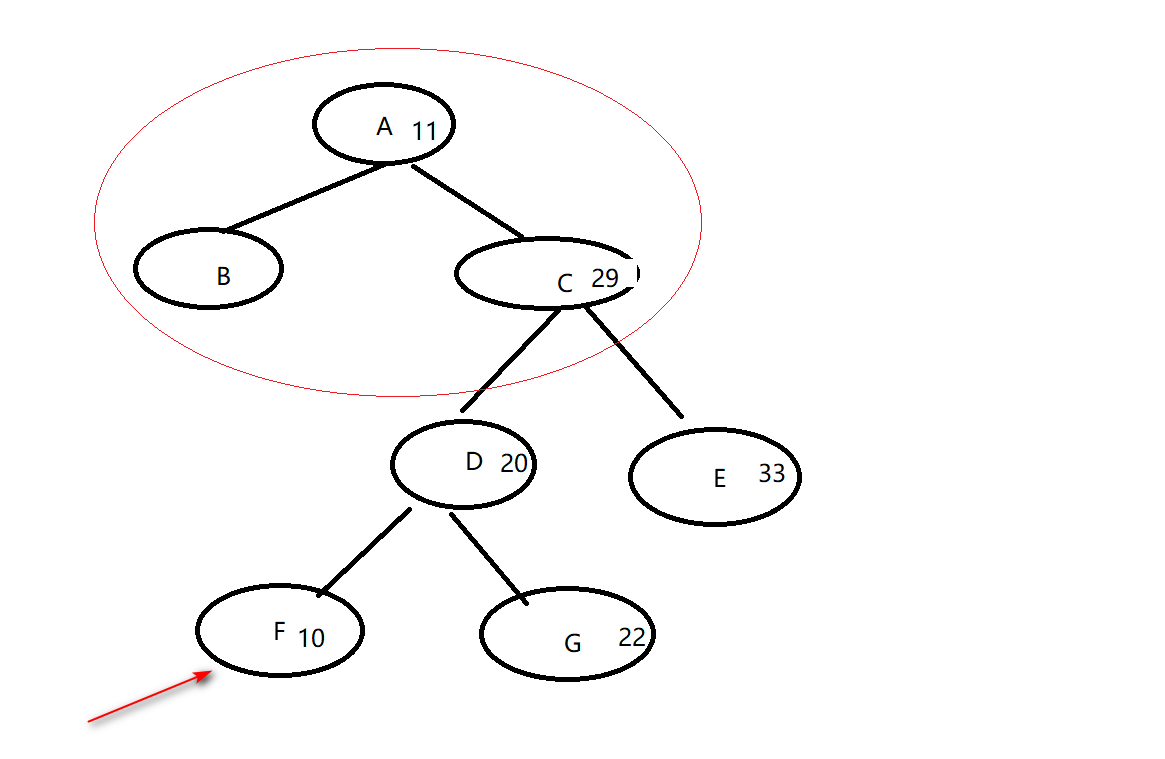
红黑树中利用左旋和右旋调整树的高度,其中

E>D>C>A>B



也就说,对于C 结点来说,C的左子树大于 c和c和向上节点,红黑树也就是利用了这个性值,才保证了左旋和右旋之后,仍然是一颗具有一些特性的二叉搜索树

会不会出现下图这种情况的红黑树呢,使得C的左子树中存在某些元素,小于c的上层树,例如F节点



答案是 错误的, 因为F节点在插入的时候就不可能通过A节点.

## 红黑树的插入

插入的步骤是,

1. 插入二叉搜索树中
2. 将插入节点标红
3. 修改这个二叉搜索树,修改的手段包括, 变色 和旋转
4. 红黑树构造完成

说明

变色和旋转的好处是 使得从根节点到叶子结点上黑色节点的数目相等,这样就保证了搜索的稳定性,n个节点的红黑树的搜索时间 稳定为log 2 n

插入的时候可能会遇到的情景

1. 插入的是根节点
2. 插入的父节点是黑节点
3. 插入的父节点是红节点

1,2 只要通过变色操作,即可,因为新插入的红节点不会影响黑节点的数目,也就保证了搜索的稳定性.

但是 如果插入的父节点是红色的,那么至少说明

1. 插入的父节点不是根节点
2. 插入的父节点不是叶子节点

这样,插入的红节点如果直接进行变色操作,那么就会影响黑节点的数目,从而影响树高,因此第3种场景就是红黑树插入操作中重点讨论的场景

参考连接

<http://www.cnblogs.com/skywang12345/p/3245399.html>



## 红黑树的删除

红黑树的删除和插入操作的思考方式一样,都是考虑特定的情境.

删除的场景有3种

1. 当个叶子节点
2. 只有左子树或者右子树的节点
3. 既有左子树又有右子树的节点

对于第三种情况,我们可以利用. 现象一下通过旋转的方式,将这个待删除节点转换为 情景 1 .2 进行操作.

转换的可行性

因为通过旋转红黑树仍然是一颗二叉搜索树,但是树的高度可能会变化,而在情景 1 ,2 种会有对于旋转之后树高的处理

参考文档

<https://www.cnblogs.com/qingergege/p/7351659.html>

<https://blog.csdn.net/qq_37169817/article/details/78880110>





## 红黑树的性质的分析

**红黑树的特性**:  
**（1）每个节点或者是黑色，或者是红色。**  
**（2）根节点是黑色。**  
**（3）每个叶子节点（NIL）是黑色。 [注意：这里叶子节点，是指为空(NIL或NULL)的叶子节点！]**  
**（4）如果一个节点是红色的，则它的子节点必须是黑色的。**  
**（5）从一个节点到该节点的子孙节点的所有路径上包含相同数目的黑节点。**

**注意**：  
(01) 特性(3)中的叶子节点，是只为空(NIL或null)的节点。  
(02) 特性(5)，确保没有一条路径会比其他路径长出俩倍。因而，红黑树是相对是接近平衡的二叉树。

### 为什么只能有两种颜色

因为红节点只是起到辅助作用,黑节点才是保证搜索稳定的

### 为什么根节点要是黑色

因为当进行旋转之后,如果遇到根节点是黑色就能停止操作了

参考文档

<https://www.zhihu.com/question/45329574/answer/109353844>



### 为什么每个叶子节点要是黑色

为了统计这个路径上节点的数目,保重每个分支上的节点数目相差不多,为了保证搜索稳定.

### 为什么红节点的子节点要全是黑色

说明红色节点起到辅助作用,黑色节点才是关键

### 为什么要路径上包含相同数目的黑节点

为了保证搜索的稳定

### 可以有父节点和叶子节点全都是黑色的红黑树吗

可以,但是只有一种情况 只有根节点,