红黑树插入删除操作图示

原理和规则：

1、每个节点要么红要么黑

2、根结点是黑色的

3、每个叶子节点是黑色的

4、红色的节点，其孩子是黑色的

4.1、不会出现连续的红色节点

4.2、红色节点的父与子都是黑色

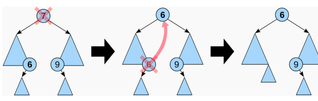
5、任意节点到叶子结点之间的黑色节点数目相同

删除操作：

着色 + 旋转 ==> 红黑树的相对平衡

这几条说明是很重要的！！

1、如果需要删除的节点有两个儿子，那么问题可以被转化为删除另一个只有一个儿子的节点的问题

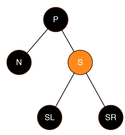


2、如果删除一个红色节点（它的儿子都是叶子节点），则直接删除

3、如果删除一个黑色节点（它的儿子是红色节点），则直接删除，然后用它的儿子顶替它，并染成黑色

因此 这里我们只讨论：删除的节点和它的儿子都是黑色的几种情况

假定要删除的节点为P，孩子节点是N，N的兄弟节点S，SL是左孩子，SR是右孩子



举例图示

如果N和P都是黑色的，则删除P会导致通过N的路径比不通过它的路径少一个黑色节点，违反性质5，因此需要重新平衡！

1、先调整（着色或旋转），再删除

2、递归的调用每种情况，直到最后待删除的节点P变为根或者是红色节点

3、N节点其实就是一个叶节点，因为我们讨论的是只有一个儿子的节点，S就是那个儿子

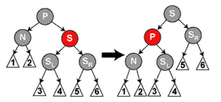
情况1：N是新的根

在删除P之后，N变成了新的根，所有路径均删除一个黑色节点

情况2：S是红色的

S是红色的和删除P有什么关系？？？
S就是那一个儿子（删除操作规则1），删完了P之后，所有经过P的路径都少一个黑节点（不经过P的路径黑节点不变），因此需要经过旋转，保证P为根的这棵子树在删除P之后各叶子结点的黑高度不变

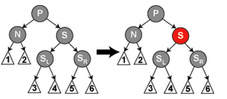
解决办法：在的父亲上做左旋，把红色兄弟变成N的祖父，接着对调N的父亲和祖父的颜色。



情况3：N、P、 S及S的儿子都是黑色的

删除P，整个子树（经过P的路径）黑高度改变，因此要做变化，保证删除后整个子树黑高度不变

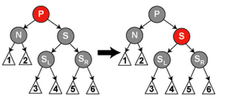
解决办法：把S兄弟染成红色，这样就变成了情况2



情况4：N、S及S的儿子都是黑色的，P是红色的

删除P，整个子树（经过P的路径）黑高度不变，但是子树没有根了，因此还得进行变化

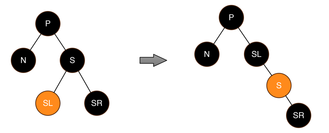
解决办法：把S和P的颜色进行交换



情况5：N、P、S及SR都是黑色的，SL是红色的

——

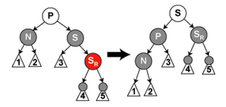
解决办法：N是P的左儿子，因此以S为轴进行右旋，并交换S和SL的颜色



情况6：S是黑色，SR是红色，N是黑色且P的左儿子

假设P是黑色的话，在子树中根节点经过N到叶节点的黑高度为2，经过SR到叶节点的黑高度为3，我们的目标就是删除且调整完毕之后还是这两个值！！

解决办法：以P为轴做左旋转，然后交换P和S的颜色



一次完整的删除推演过程