红黑树（二）：删除

海纳
公众号HinusWeekly

已经有读者告诉我说，不要再讲删除了，肯定看不懂。但我觉得，其实真没有那么难。所以删除我还是要讲一下的。看完了你就会发现，其实真的不难。
红黑树的删除确实复杂。侯捷的那本《STL源码解析》都跳过了这一节。《算法导论》里倒是没有跳过，但是我认为讲得不是很好。今天我不自量力，也来挑战一下，看能不能讲清楚了。

如果需要删除的结点有两个孩子，我们的做法是找到这个结点的中序后继，将后继结点中的数据拷贝至待删除结点，然后删除后继结点。而后继结点必然最多只有一个子结点，这样我们就把删除两个孩子的结点转为删除一个孩子的结点。在删除的过程中，我们只关心树是否仍然保持红黑树的性质，数据是否组织正确，而不关心这个结点是最初要删除的结点还是我们从中复制出值的那个结点。

接下来，我们只讨论删除只有一个儿子的结点，如果它两个儿子都为空，即均为叶子，我们任意将其中一个看作它的儿子。这里，体现出来，在红黑树里特别指定叶子结点为NIL结点的作用，NIL结点经常可以充当正常结点使用以使得算法的表达更加容易。

如果我们删除一个红色结点，它的父亲和儿子一定是黑色的。所以我们可以简单的用它的黑色儿子替换它，并不会破坏属性2和3。通过被删除结点的所有路径只是少了一个红色结点，这样可以继续保证属性4。另一种简单情况是在被删除结点是黑色而它的儿子是红色的时候。如果只是去除这个黑色结点，用它的红色儿子顶替上来的话，会破坏属性4，但是如果我们重绘它的儿子为黑色，则曾经通过它的所有路径将通过它的黑色儿子，这样可以继续保持属性4。这是比较简单的两种情况，我们不再讨论。

现在，需要详细讨论的就是如果待删除结点和它的子结点都是黑色的时候，这种比较复杂的情况。有些参考资料上为了表达形式上的简洁，把要讨论的情况分成了四类或者五类，而我认为，把情况细分为六类是比较合理的，这样更利于理解。而且不同于其他的讲解的地方是，我在这里调整了这六种情况的排序。先讲解最简单的，最后讲最复杂的，而不是追求形式上完善，这一点希望读者能够理解。

先来约定涉及到的结点的名称。我们先用待删除结点的孩子代替待删除结点，并且记这个孩子为N，记它的新的父结点为P，它的兄弟结点，也就是父结点的另外一个孩子为S， 记S的左孩子为 S_L ，记S的右孩子为 S_R 。

情况一。N是新的根。在这种情况下，我们就做完了。我们从所有路径去除了一个黑色节点，而新根是黑色的，所有属性都保持着。

了这条路径上删除的黑色结点。所以红黑树又重新达到了平衡。

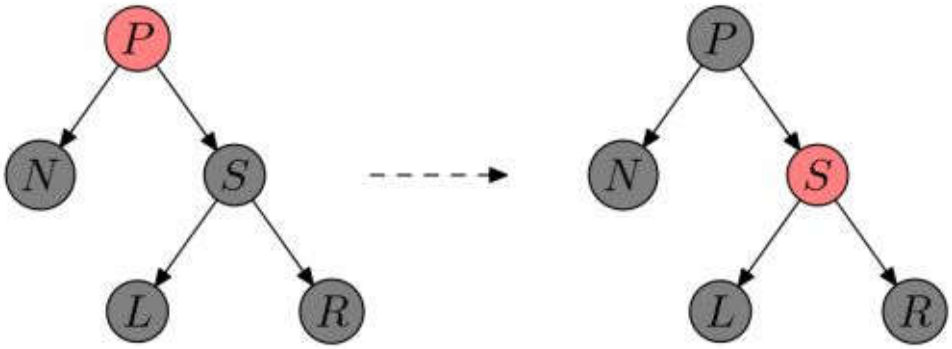

图 2.17: 第二种情况

情况三。S是黑色，S的右儿子是红色，而N是它父亲的左儿子。在这种情况下我们在N的父亲上做左旋转，这样S成为N的父亲和S的右儿子的父亲。我们接着交换N的父亲和S的颜色，并使S的右儿子为黑色。子树在它的根上的仍是同样的颜色，所以属性3 没有被违反。但是，N现在增加了一个黑色祖先: 要么N的父亲变成黑色，要么它是黑色而S被增加为一个黑色祖父。所以，通过N的路径都增加了一个黑色节点。此时，如果一个路径不 通过N，则有两种可能性:

1. 它通过N 的新兄弟。那么它以前和现在都必定通过S 和N 的父亲，而它们只是交换了颜色。所以路径保持了同样数目的黑色节点。
2. 它通过N 的新叔父，S 的右儿子。那么它以前通过S、S 的父亲和S的右儿子，但是现在只通过S，它被假定为它以前的父亲的颜色，和 S 的右儿子，它被从红色改变为黑色。合成效果是这个路径通过了同样数目的黑色节点。

在任何情况下，在这些路径上的黑色节点数目都没有改变。所以我们恢复了属性4。在示意图中的白色节点可以是红色或黑色，但是在变换前后都必须指定相同的颜色

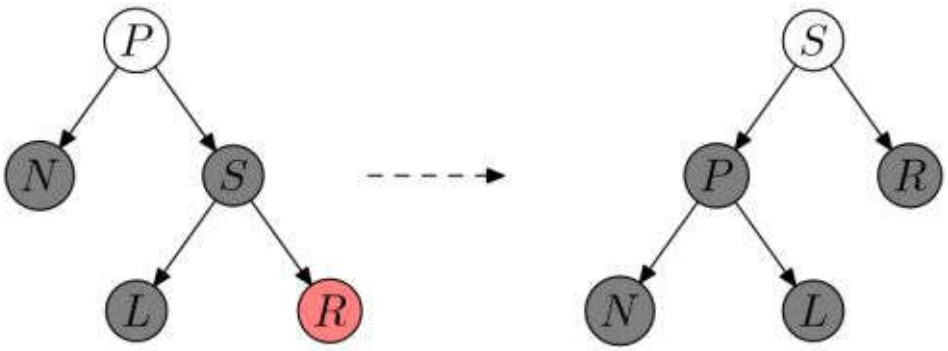

图 2.18: 第三种情况

情况四。S是黑色，S的左儿子是红色，S的
我们在S上做右旋转，这样S的左儿子成为S

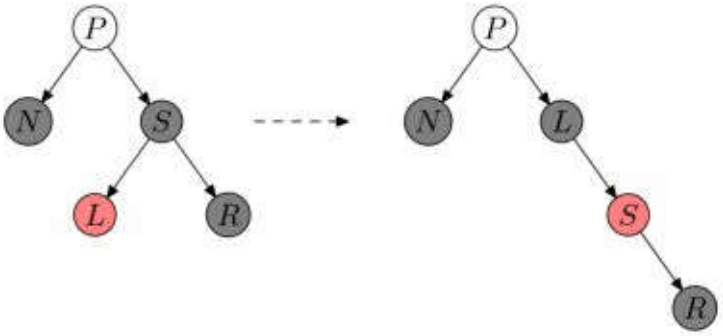

图 2.19: 第四种情况

情况五。S是红色。在这种情况下我们在N的父亲上做左旋转，把红色兄弟转换成N的 祖父。我们接着对调N的父亲和祖父的颜色。尽管所有的路径仍然有相同数目的黑色节点，现在N有了一个黑色的兄弟和一个红色的父亲，所以我们可以接下去按二、三或四情况来处 理。N它的新兄弟SL是黑色，因为它未旋转前是红色S的一个儿子。

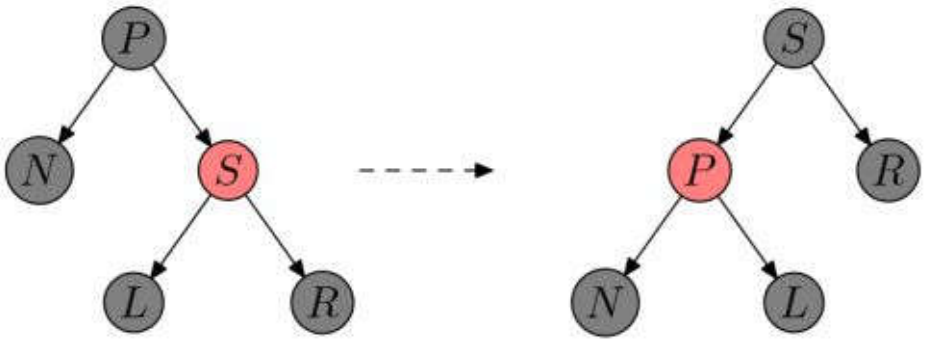

图 2.20: 第五种情况

情况六。N的父亲、S和S的儿子都是黑色的。在这种情况下，我们简单的重绘S为红 色。结果是通过S的所有路径，它们就是以前不通过N的那些路径，都少了一个黑色节点。 因为删除N的初始的父亲使通过N的所有路径少了一个黑色节点，这使事情都平衡了起来。 但是，通过P的所有路径现在比不通过P的路径少了一个黑色节点，所以仍然违反属性3。要修正这个问题，我们要从情况一开始，在P上做重新平衡处理。

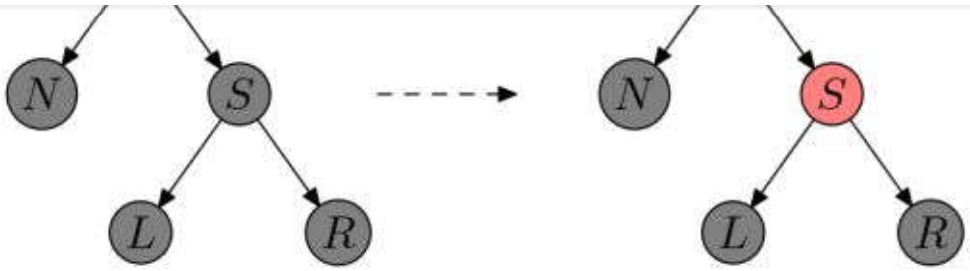

图 2.21: 第六种情况

好了。今天就到这里了。大家自己画画看。下节课，我们讲具体的代码实现。

编辑于 2017-02-23

Java 数据结构 红黑树

文章被以下专栏收录

进击的Java新人 进入专栏

推荐阅读

数据结构：红黑树的删除

Cailiang

红黑树深入剖析及Java实现

红黑树是平衡二叉查找树的一种。为了深入理解红黑树，我们需要从二叉查找树开始讲起。BST二叉查找树（Binary Search Tree，简称BST）是一棵二叉树，它的左子节点的值比父节点的值要小，右节...

美团技术团... 发表于美团技术博...

码

看

首发于
进击的Java新人

写下你的评论...

烟雨红尘醉相思 1 年前

一脸懵逼

2

Patrick Wang 1 年前

情况5中到达N的黑色节点数是不是少了一个？

赞

海纳 (作者) 回复 Patrick Wang 1 年前

情况5要先转变成情况2，他不是直接删除N的。n的父节点和兄弟节点对换一下颜色就仍然平衡了

赞 查看对话

叶是苍 1 年前

谢谢讲解~

赞

slg 1 年前

情况6中，s变成红色后是不是应该进入情况5？怎么从情况一开始了？

赞

知乎用户 1 年前

谢谢~

赞

大兔崽子 1 年前

情况六和情况二不是重复了么，具体实现的时候调整过程是一个循环，跳出循环的条件就是待调整节点为红节点或为根节点，因为待调整节点为红节点是直接置黑就可以了，第二种情况按第六种情况的处理方法处理的时候，待调整节点从N变为P，下一步就是跳出循环了，没必要单独分开说，这也是算导上只说了情况六的原因

1

小吉 1 年前

想问的一点是，到底哪个是要删除的节点？

赞

待后继结点中的数据拷贝至待删除结点，然后删除后继结点。而后继结点必然最多只有一个子结点。为什么后续节点必然最多只有一个子节点，有两个不行吗?

赞

柳木澄泉 1 年前

“我们先用待删除结点的孩子代替待删除结点，并且记这个孩子为N，记它的新的父结点为P。”这里应该有图，语言的表述能力是不够的。这里的“它”指的前面的“待删除节点”还是“这个孩子”？我感觉看到这里理解不了影响后续阅读。

赞

Jitre 回复 柳木澄泉 1 年前

因为他是中序的后继

赞 查看对话

joejoe 12 个月前

终于懂了啊。。算法导论看不懂

赞

BeBornAgain 9 个月前

关键在于搞清楚每一条链路上的黑色节点值为多少，在对黑色节点不够的链路调整时，要保证有足够的红色节点用于平衡黑色节点的值。。。

赞

imtian 回复 小吉 8 个月前

维基百科红黑树说的比较清楚，建议去看看。我也写了一篇关于红黑树的文章，对大家容易产生误解的地方进行了说明，有兴趣可以看看，地址在这：segmentfault.com/a/1190...

3 查看对话

杨伟华 5 个月前

逻辑清晰，wiki百科上也是这样

赞