Homework 5: RB-Tree Implementation

赵楷越 522031910803

1 代码正确性证明

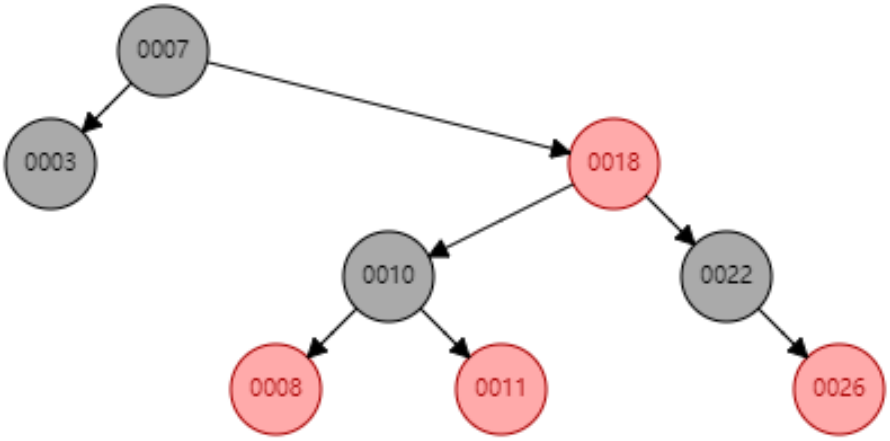

Figure 1: 插入 7，3，18，10，22，8，11，26 后的红黑树

```
ubuntu2204@horizon22:/mnt/hgfs/share/hw5/rbtree$ ./test
Inorder traversal of the constructed tree:
3 BLACK
7 BLACK
8 RED
10 BLACK
11 RED
18 RED
22 BLACK
26 RED
```

Figure 2: 程序运行结果的截图

2 删除操作分析

这部分按照了 hw4 中的 Part 2 第一问的要求，解释了删除后采取的失衡恢复操作，是如何使得红黑树依然维持其性。具体分成了两种情况、其中这两种情况又进一步细分成了共七种情况。对应的轴对称情况与之类似，不在此赘述。（注：本部分的作图参考了此篇文章，其中绿色标记出的为待修复的失衡节点，蓝色标记出的是不影响红黑树满足局部性质的节点）

2.1 失衡节点的父节点是红色

对父节点做左旋：原来父节点 2 左侧路径黑色节点少于右侧路径，经过对父节点 2 的左旋之后，使得从局部根节点往节点 1 的路径上多一个黑色节点，完成失衡修复。

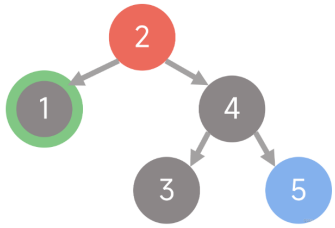

Figure 3: 兄弟的左孩子是黑色

将父亲染成黑色：使得从局部根节点往节点 1 的路径上多一个黑色节点，进行失衡修复。但是也导致了右侧路径上多了一个黑色节点。

将兄弟染成红色：使得右侧路径上减少一个黑色节点、与原来相同。但是产生了节点 3 和 4 的双红问题。

进行双红修正：使红黑树满足其原有性质。

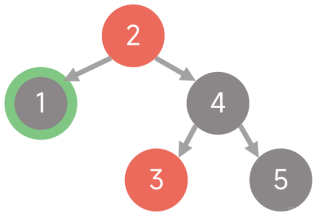

Figure 4: 兄弟的左孩子是红色，兄弟的右孩子是黑色

将父亲染成黑色：使得从局部根节点往节点 1 的路径上多一个黑色节点，进行失衡修复。但是也导致了右侧路径上多了一个黑色节点。

将兄弟染成红色：减少右侧路径上的一个黑色节点个数、但是让右侧路径上有三个组合的红色节点，不满足性质。

将兄弟的右孩子染成黑色：将三个组合的红色节点转化为双红问题。

对父亲做左旋：进行双红问题的处理。

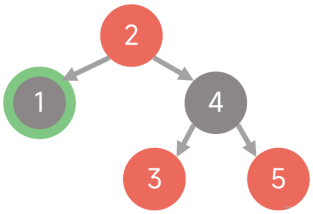

Figure 5: 兄弟节点的两个孩子都是红色的

2.2 失衡节点的父节点是黑色

将兄弟节点染红：因为此时怎样都无法修复、因此在此先让局部满足红黑树的性质。

将父节点 2 标为新的失衡节点：求助爷爷节点，类似于 b 树中的下层下溢引发上层下溢。

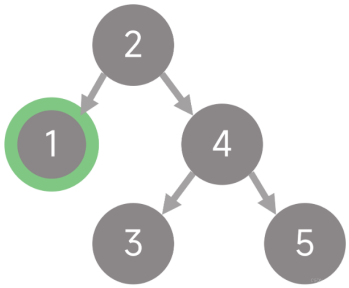

Figure 6: 兄弟节点为黑，兄弟的孩子都是黑色的

对父节点 2 做左旋：使得从局部根节点往节点 1 的路径上多一个黑色节点，进行失衡修复。但也导致了右侧路径上的黑色节点减少了一个。

将兄弟节点 4 的右孩子 5 染黑：使得右侧路径上的黑色节点增加一个，恢复其原有性质。

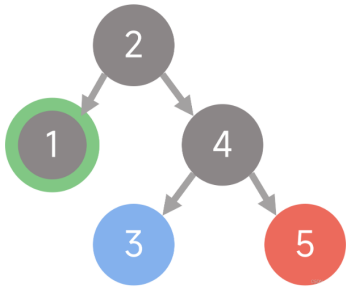

Figure 7: 兄弟节点为黑，兄弟节点的右孩子为红

将兄弟的左孩子 4 染成黑色：要使得从局部根节点往节点 1 的路径上多一个黑色节点，我们应该想办法让父亲节点 2 有一个新的黑色的父亲节点，此处染黑节点 4 操作，使得有足够的黑色节点填充。

对兄弟做右旋：调整局部的结构，为了使得能从局部根节点往节点 1 的路径上多一个黑色节点。

对父亲做左旋：进行最终旋转，保证了左侧路径上的黑色节点多了一个，而右侧路径上的黑色节点个数不变。

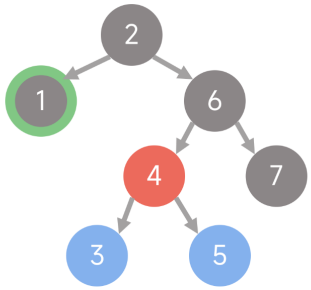

Figure 8: 兄弟节点为黑色，兄弟节点的左孩子为红色，兄弟节点的右孩子为黑色

对父亲做左旋，并将将兄弟染成黑色：尝试使得从局部根节点往节点 1 的路径上多一个黑色节点，但也导致节点 3 上的路径多了一个黑色节点。

将父亲染成红色：恢复节点 3 路径上的黑色节点个数，转化局部失衡节点修复关系。此时 1 是失衡节点、2 是红色节点、3 是黑色节点。

按照失衡节点 1 的父节点 2 是红色节点（先前提到的三种方法）进行处理：进行最终失衡修复。

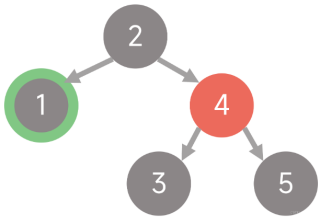

Figure 9: 兄弟节点为红色

3 对顺序和乱序插入的实验结果统计与分析

插入的数字序列是 0-9999（共 10000 个元素），先后进行顺序插入和乱序插入。顺序插入采用从小到大依次插入的方式；乱序插入采用对这些元素随机排序后再插入的方式。共进行了一次顺序插入，10000 次乱序插入。统计了一共发生了多少次失衡恢复，失衡恢复中一共执行了多少次染色，多少次旋转。实验结果如下表所示：

	失衡恢复次数	染色次数	左旋次数	右旋次数	旋转次数
顺序插入	9998	49865	9976	0	9976
乱序插入的平均值	6284	23143	2911	2910	5821

Table 1: 实验结果

根据实验结果，可以看出顺序插入和乱序插入对红黑树的插入操作产生了不同的影响，具体分析比较两种不同的插入方式的效果如下：

顺序插入：

- **失衡恢复次数较多：**共计 9998 次失衡恢复，说明在顺序插入的情况下，红黑树更容易发生失衡。
- **染色次数较多：**共计 49865 次染色操作，说明在顺序插入过程中，需要频繁地对节点进行染色操作以维持红黑树的性质。
- **旋转次数较多：**共计 9976 次旋转操作，说明在顺序插入过程中，需要进行大量的旋转操作来调整树的结构。

乱序插入：

- **失衡恢复次数较少：**平均 6284 次失衡恢复，因此相较于顺序插入，乱序插入时失衡恢复的次数显著减少，说明乱序插入对红黑树的平衡性有一定的改善作用。
- **染色次数较少：**平均 23143 次染色操作，相对于顺序插入，因此乱序插入时需要的染色操作较少，表明乱序插入能够减少对节点颜色的频繁修改。
- **旋转次数较少：**平均 2911 次左旋和 2910 次右旋，平均 5821 次旋转操作，因此乱序插入时需要的旋转操作相对较少，说明乱序插入对红黑树的结构调整影响较小。

综上所述，乱序插入相比于顺序插入对红黑树的插入操作更有利，可以减少失衡恢复次数、染色次数以及旋转次数，从而提高插入效率和降低对红黑树结构的频繁调整，使得红黑树能够更好地保持平衡。