hw6

> 13.1-3，13.3-4；13.4-3；14.1-5，14.2-2，14.3-3；

13.1-3

13.1-3 定义一棵**松弛红黑树**(relaxed red-black tree)为满足红黑性质 1、3、4 和 5 的二叉搜索树。换句话说，根结点可以是红色或是黑色。考虑一棵根结点为红色的松弛红黑树 T。如果将 T 的根结点标为黑色而其他都不变，那么所得到的是否还是一棵红黑树？

满足条件。因为没有引入红色节点，所以条件4仍然满足。由于从根到叶子的每条路径中都有根，但没有其他节点。条件5也会得到满足，因为只改变从根开始的路径中的黑色节点数量。所有这些数量都会增加1，因此它们都是相等的。条件3很容易保持，因为没有新的叶子被引入。条件1也很容易保持，因为只有一个节点被改变，并且它没有被改变成某种神秘的第三种颜色。

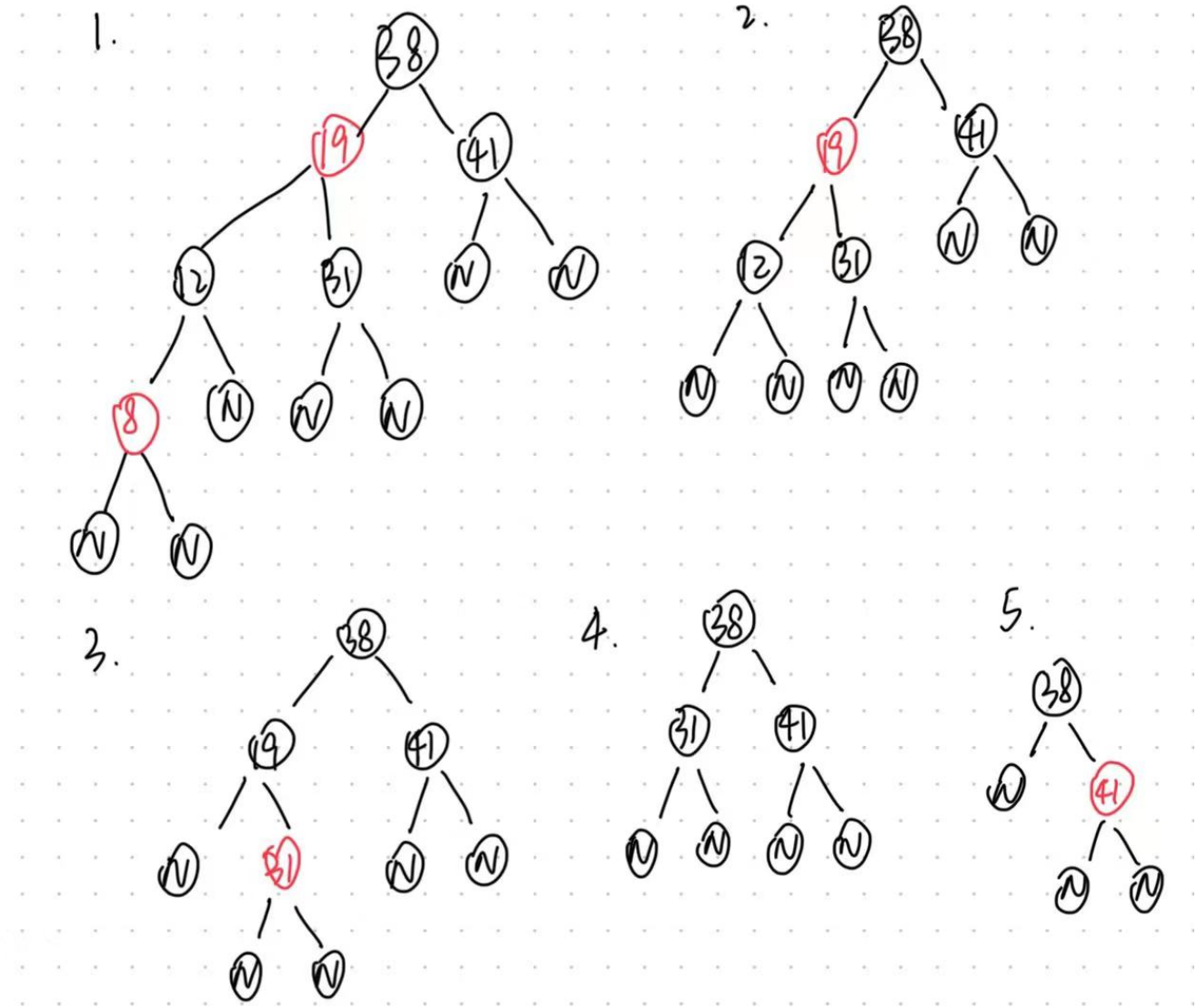
13.3-4

13.3-4 Teach 教授担心 RB-INSERT-FIXUP 可能将 $T.nil.color$ 设为 RED，这时，当 z 为根时，第 1 行的测试就不会让循环终止。通过讨论 RB-INSERT-FIXUP 永远不会将 $T.nil.color$ 设置为 RED，来说明这位教授的担心是没有必要的。

首先注意到，RB-INSERT-FIXUP 只有在一个节点的子节点已经是红色时才会修改它，所以永远不会修改一个已经被设置为 T.nil 的子节点。只需要检查根节点的父节点是否会被设置为红色。由于根节点和其父节点都默认是黑色的，如果 z 的深度小于2，while循环就会被打破。最多只修改 z 上方两个层级的节点的颜色，所以只需要担心 z 在深度2的情况。在这种情况下，有可能会将根节点修改为红色，但这在第16行中已经处理过了。当 z 被更新时，它要么是根节点，要么是根节点的子节点。无论如何，根节点和其父节点仍然是黑色的，所以while的条件不满足，使得修改 T.nil 为红色成为不可能。

13.4-3

13.4-3 在练习 13.3-2 中，将关键字 41、38、31、12、19、8 连续插入一棵初始的空树中，从而得到一棵红黑树。请给出从该树中连续删除关键字 8、12、19、31、38、41 后的红黑树。



14.1-5

14.1-5 给定 n 个元素的顺序统计树中的一个元素 x 和一个自然数 i，如何在 $O(\lg n)$ 的时间内确定 x 在该树线性序中的第 i 个后继？

期望的结果是 OS-SELECT$(T, OS-RANK(T, x)+i)$。其运行时间为 $\mathrm{O(h)}$，根据红黑树的性质，这等于 $O(\lg(n))$。

14.2-2

14.2-2 能否在不影响红黑树任何操作的渐近性能的前提下，将结点的黑高作为树中结点的一个属性来维护？说明如何做，如果不能，请说明理由。如何维护结点的深度？

由于一个节点的黑高度仅取决于其子节点的黑高度和颜色，因此定理14.1意味着我们可以维护该属性而不影响其他红黑树操作的渐近性能。但维护节点的深度则不是如此。如果删除了一棵树的根节点，可能需要更新 $O(n)$ 个节点的深度，这使得DELETE操作的渐近速度比以前慢。

14.3-3

14.3-3 请给出一个有效的算法，对一个给定的区间 i，返回一个与 i 重叠且具有最小低端点的区间；或者当这样的区间不存在时返回 $T.nil$。

考虑常用的区间搜索方法。但在这里，不是在检测到重叠时立刻中断循环，而是持续追踪最新发现的重叠区间。会持续这个循环直到遇到T.nil为止。此时，返回最近发现的重叠区间。由于在存在重叠的情况下，搜索总是倾向于向左侧移动，而左侧的子节点具有较小的左端点，因此这个方法能确保找到的是具有最小左端点的重叠区间。