红黑树

讲义

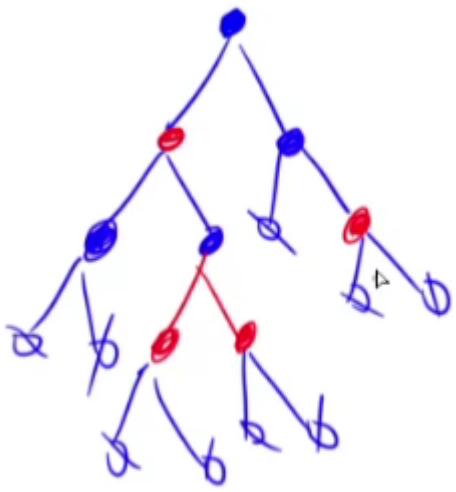
```
第21讲 红黑树和并查集
1. 红黑树
    1.1 定义
        (1) 结点是红色或黑色。
        (2) 根结点是黑色。
        (3) 所有叶子都是黑色。（叶子是NIL结点）
        (4) 每个红色结点的两个子结点都是黑色。（从每个叶子到根的所有路径上不能有两个连续的红色结点）
        (5) 从任一节结点其每个叶子的所有路径都包含相同数目的黑色结点。
    1.2 性质
        从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。
    1.3 平衡操作
        1.3.1 插入
            1.3.1.1 被插入的节点是根节点。
                直接把此节点涂为黑色。
            1.3.1.2 被插入的节点的父节点是黑色。
                什么也不需要做。
            1.3.1.3 被插入的节点的父节点是红色。
                [1] 当前节点的祖父节点的另一个子节点（叔叔节点）也是红色。
                    (1) 将"父节点"设为黑色。
                    (2) 将"叔叔节点"设为黑色。
                    (3) 将"祖父节点"设为"红色"。
                    (4) 将"祖父节点"设为"当前节点"(红色节点)；即，之后继续对"当前节点"进行操作。
                [2] 叔叔节点是黑色，且当前节点是其父节点的右孩子
                    (1) 将"父节点"作为"新的当前节点"。
                    (2) 以"新的当前节点"为支点进行左旋。
                [3] 叔叔节点是黑色，且当前节点是其父节点的左孩子
                    (1) 将"父节点"设为"黑色"。
                    (2) 将"祖父节点"设为"红色"。
                    (3) 以"祖父节点"为支点进行右旋。
        1.3.2 删除
            1.3.2.1 x指向一个"红+黑"节点。
                将x设为一个"黑"节点即可。
            1.3.2.2 x指向根。
                将x设为一个"黑"节点即可。
            1.3.2.3
```

基础课报名人数 | 讲义.txt | 回复

```
391         (3) 所有叶子都是黑色。（叶子是NIL结点）
395         (4) 每个红色结点的两个子结点都是黑色。（从每个叶子到根的所有路径上不能有两个连续的红色结点）
396         (5) 从任一节结点其每个叶子的所有路径都包含相同数目的黑色结点。
397     1.2 性质
398         从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。
399     1.3 平衡操作
400         1.3.1 插入
401             1.3.1.1 被插入的节点是根节点。
402                 直接把此节点涂为黑色。
403             1.3.1.2 被插入的节点的父节点是黑色。
404                 什么也不需要做。
405             1.3.1.3 被插入的节点的父节点是红色。
406                 [1] 当前节点的祖父节点的另一个子节点（叔叔节点）也是红色。
407                     (1) 将"父节点"设为黑色。
408                     (2) 将"叔叔节点"设为黑色。
409                     (3) 将"祖父节点"设为"红色"。
410                     (4) 将"祖父节点"设为"当前节点"(红色节点)；即，之后继续对"当前节点"进行操作。
411                 [2] 叔叔节点是黑色，且当前节点是其父节点的右孩子
412                     (1) 将"父节点"作为"新的当前节点"。
413                     (2) 以"新的当前节点"为支点进行左旋。
414                 [3] 叔叔节点是黑色，且当前节点是其父节点的左孩子
415                     (1) 将"父节点"设为"黑色"。
416                     (2) 将"祖父节点"设为"红色"。
417                     (3) 以"祖父节点"为支点进行右旋。
418         1.3.2 删除
419             1.3.2.1 x指向一个"红+黑"节点。
420                 将x设为一个"黑"节点即可。
421             1.3.2.2 x指向根。
422                     (2) 设置"x的父节点"为"新的x节点"。
423                 [3] x的兄弟节点是黑色；x的兄弟节点的左孩子是红色，右孩子是黑色的。
424                     (1) 将x兄弟节点的左孩子设为"黑色"。
425                     (2) 将x兄弟节点设为"红色"。
426                     (3) 对x的兄弟节点进行右旋。
427                     (4) 右旋后，重新设置x的兄弟节点。
428                 [4] x的兄弟节点是黑色；x的兄弟节点的右孩子是红色的，x的兄弟节点的左孩子任意颜色。
429                     (1) 将x父节点颜色 赋值给 x的兄弟节点。
430                     (2) 将x父节点设为"黑色"。
431                     (3) 将x兄弟节点的右子节设为"黑色"。
432                     (4) 对x的父节点进行左旋。
433
434
435 2. 并查集
436     2.1 定义
437         用森林维护集合关系，支持合并、查询等操作。
```

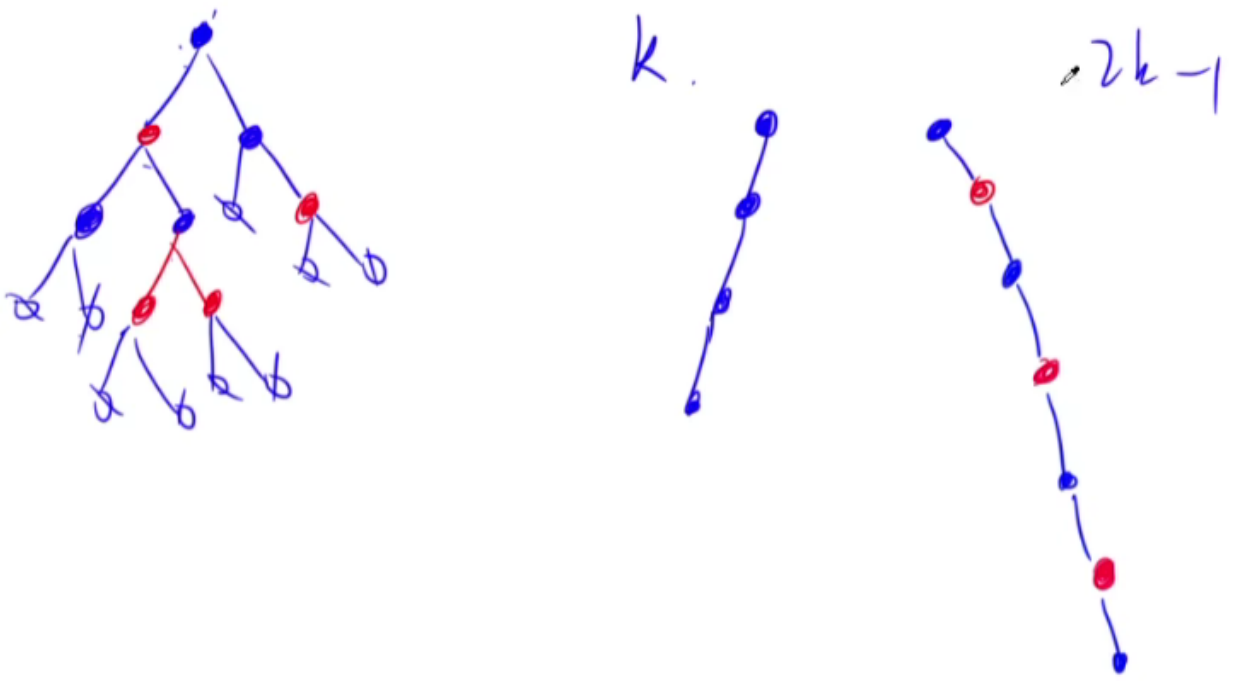
竞赛常用：splay，treap，fhqtreap

红黑树



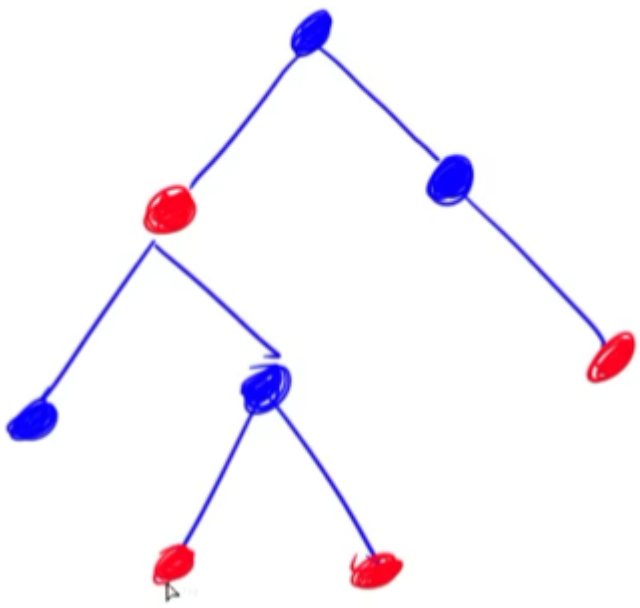
性质（理解补充）：

- 最短路径，最长路径。期望高度O(logN)，红黑树所有操作都是O(logN)级别

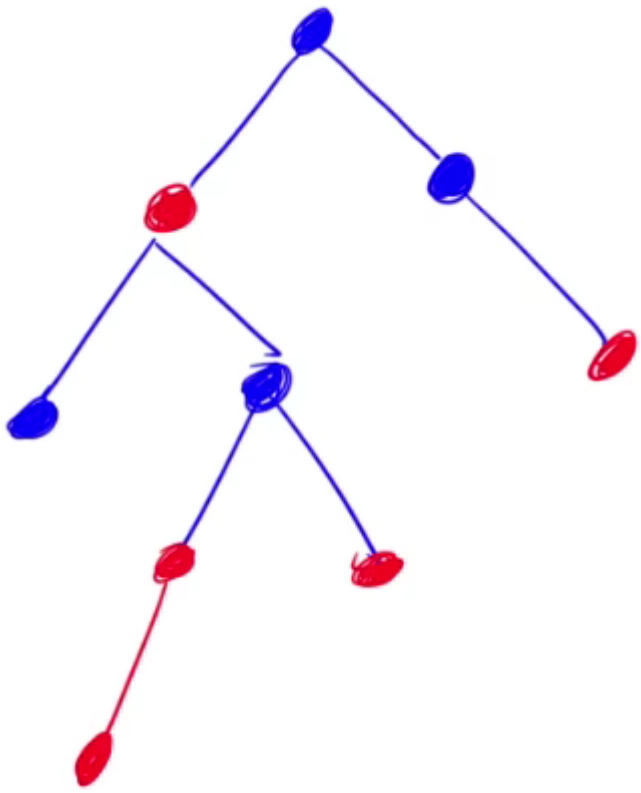


操作：

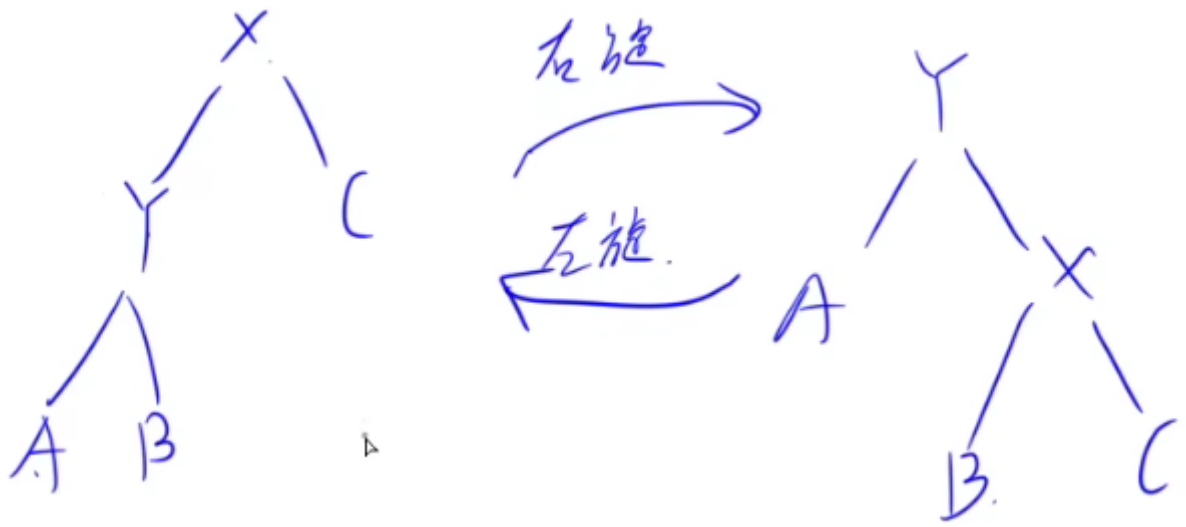
- 插入



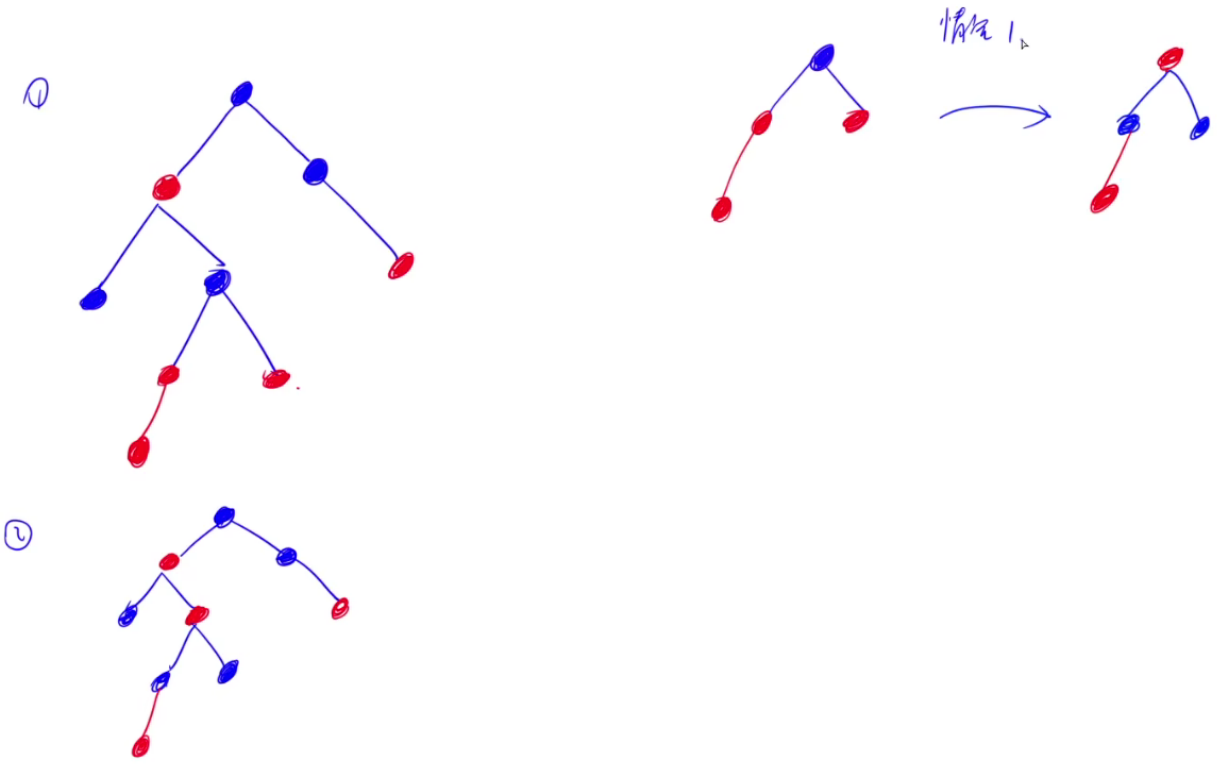
1. 先按照BST的规则插入



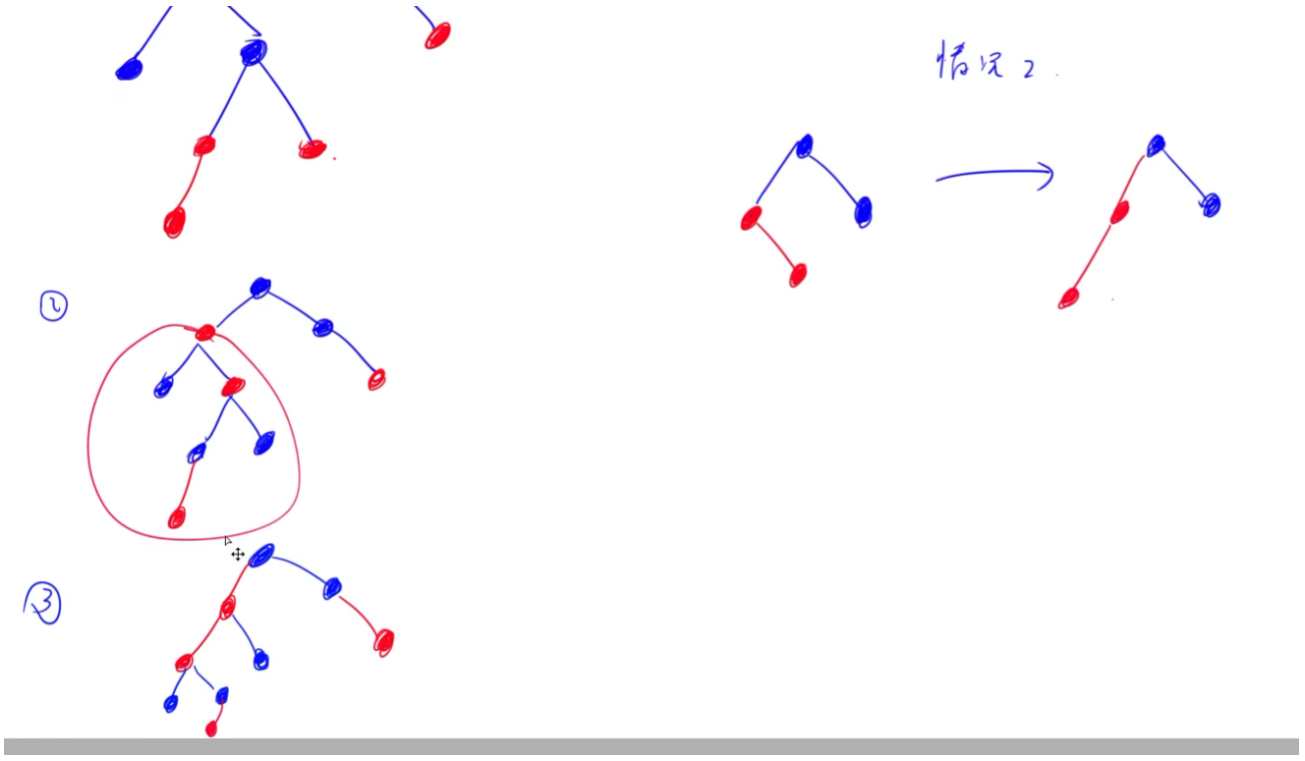
1. 左右旋



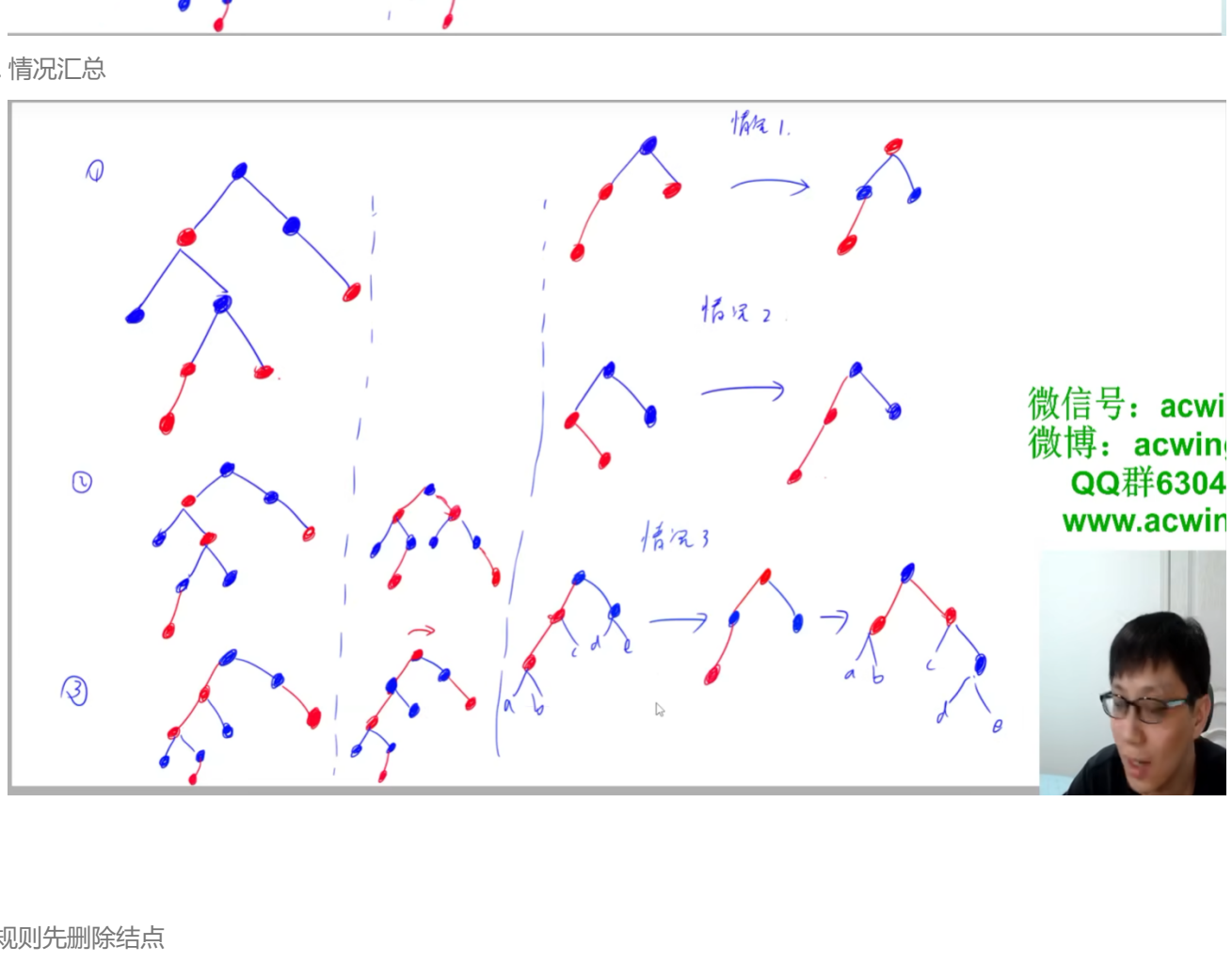
2. 维护红黑树的性质
    1. 如果插入结点是根节点，直接把该节点变为黑色
    2. 如果插入的结点的父结点是黑色，则当前结点变为红色，其他不做操作
    3. 如果插入的结点父结点是红色
        1. 情况1（需进一步递归操作）



2. 情况2 （转化为情况3）

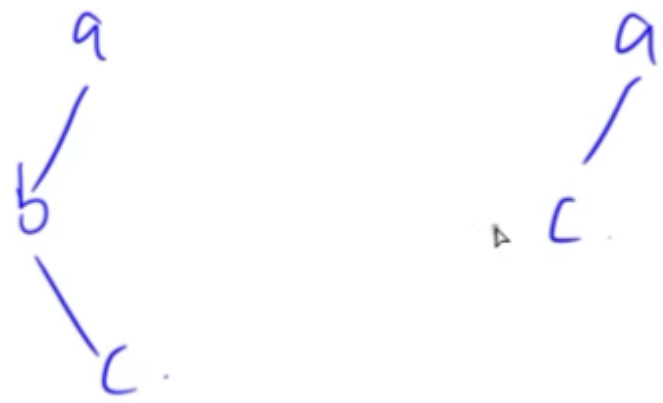


3. 情况3（最终满足红黑性质）



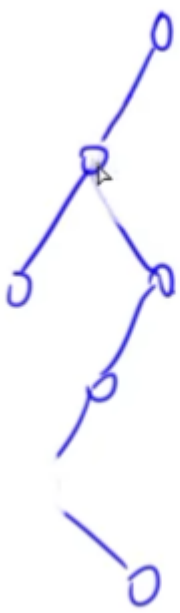
4. 情况汇总

- 删除
    1. 按照BST规则先删除结点
        1. 情况1



2. 情况2

3. 情况3

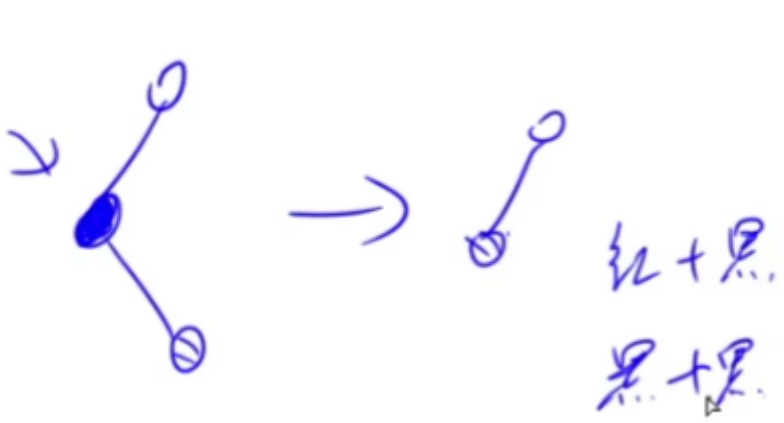


2. 维护红黑树的性质

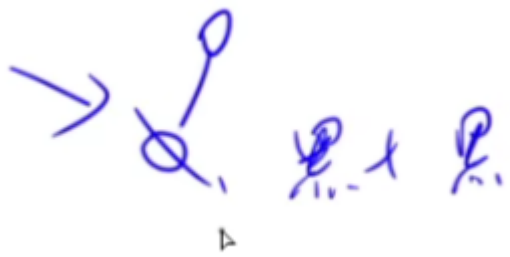
   如果删除的点是红色点，则没有影响，无需操作

   删除的点a是**黑色**点，则给替代a位置点b的颜色多赋值一个**黑色**(留下a点的颜色)，即b点的颜色是“红+黑”或者“黑+黑”。

   - 如果删除的点a至少有一个左右儿子，则最终一定会转换成以下情况



   - 如果删除的点a没有左右儿子，由于红黑树多维护了Nil空结点c，且空结点为黑色，所以将a的黑色累加到空结点c（黑+黑）上。



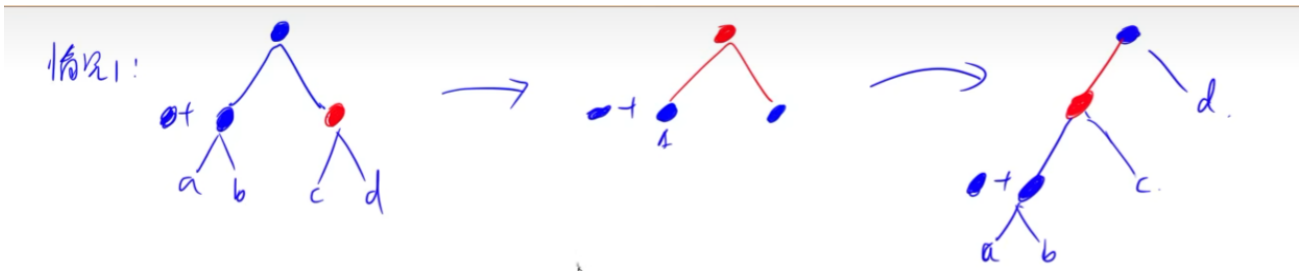
   将替代删除的a点位置的点设为x点

   1. 如果x = 红+黑，则将x = 黑

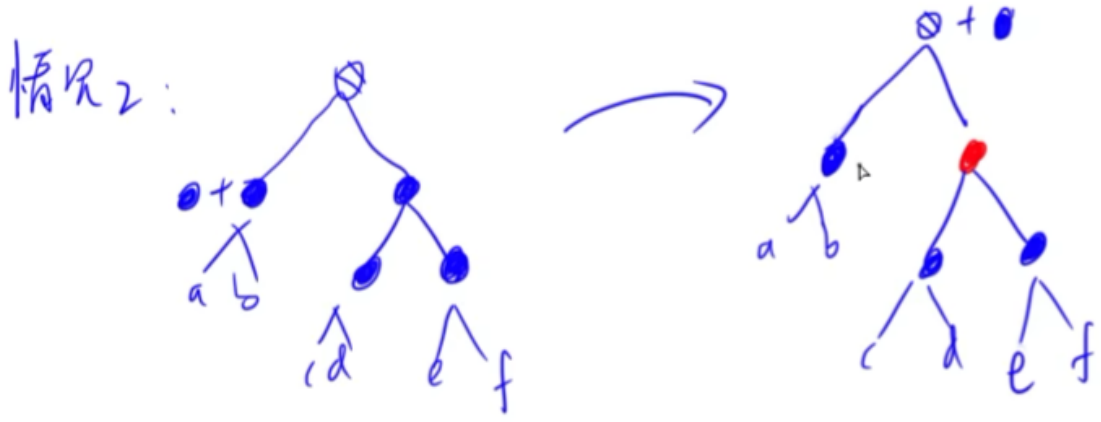
2. 如果x是根节点，则x = 黑+黑，且root结点去掉一个黑色仍满足性质5，故将x = 黑，即所有路径都去掉一个黑色点

3. 如果不是以上情况（当前结点”黑+黑“，且不是根节点）

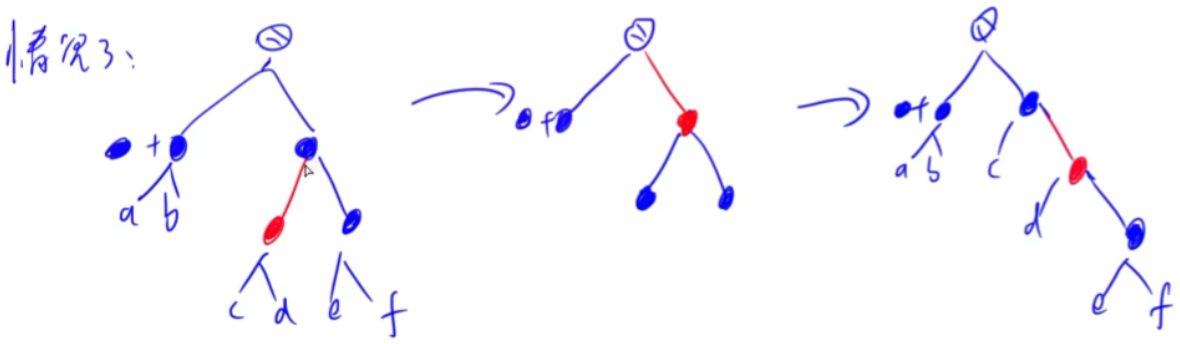
1. 情况1（将兄弟结点变为黑色->情况2，3，4，最多执行O(logN)次）



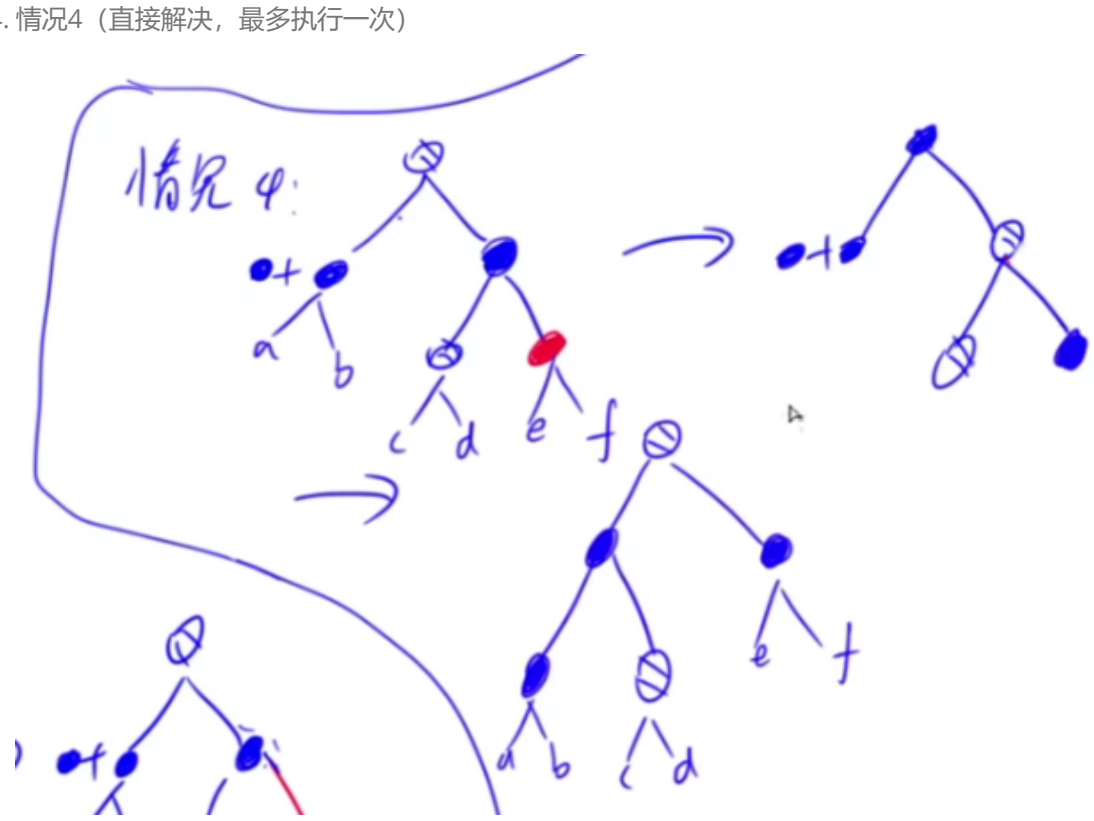
2. 情况2（将要处理的结点往上移一层->不断递归至根节点解决，2最多执行O(logN)次，同时顺带可能执行让1执行O(logN)次）



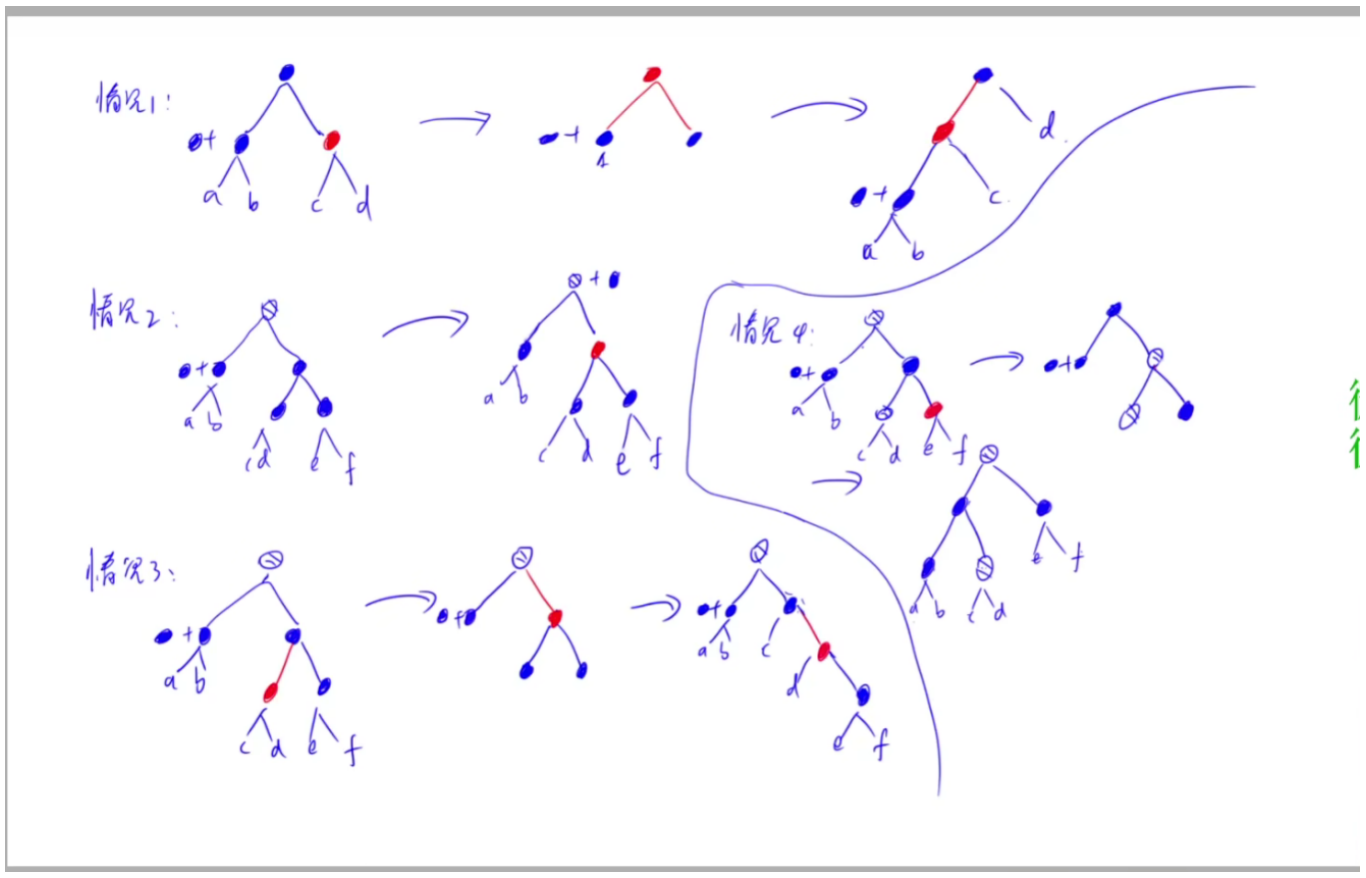
3. 情况3 （将兄弟结点右儿子变为红色->情况4，最多执行1次）



4. 情况4（直接解决，最多执行一次）



因此整个操作的时间复杂度是O(logN)级别



- 修改（先删除，后插入）