#! https://zhuanlan.zhihu.com/p/598626361

算法学习笔记

红黑树

性质：

- 每个节点要么是红色要么是黑色
- 根节点和叶节点是黑色
- 如果一个节点是红色那么它的孩子是黑色
- 任意节点到叶子节点的树链中含有相同数量的黑色节点

最后一条的最优情况是两条路径上节点数一样，最差是有一条路径上的节点数是另一条的两倍

关于红黑树的思考

红黑树与AVL树

红黑树本质上是对二叉搜索树的一种优化，目的是保证树的高度，使整棵树不会太失衡。与AVL树不同的是，红黑树认为的平衡是两个分支上的节点数不会相差一倍以上。正是基于这一点，保证了红黑树的读和写都会是log的级别。

由于红黑树的平衡要求比AVL树要低，所以在插入或者删除节点的时候，所需要的左旋和右旋次数也要相应的少很多（据说最多三次就可以达到平衡），所以在频繁写入的情境下红黑树的性能要比AVL树更加优秀。

红黑树是如何保证树的高度的

红黑树维护树的高度不会过度失衡的手段就是节点的颜色，特别是红色节点。根据红黑树的性质，我们容易发现，红色节点的数量至多等于黑色的节点。而根据最后一条，我们在忽略红色节点的情况下这是一棵AVL树，所以整个树的高度最多也就是黑色节点的两倍。而黑色节点根据AVL树的性质可以知道至多也就是log级别的，这就保证了树的高度

插入流程：

父节点	叔节点	类型	操作
黑	-	-	无需操作
红	红	-	父、叔都变黑，祖父变红。祖父变成当前节点，递归这个操作
红	黑	左左	右旋+变色

父节点	叔节点	类型	操作
红	黑	右右	左旋+变色
红	黑	左右	左旋+右旋+变色
红	黑	右左	右旋+左旋+变色

插入流程：

- 按照二叉搜索树的方法寻找插入的节点在哪里
  - 如果有符合要求的节点则直接该点的cnt++
  - 否则在他的左儿子或者右儿子插入该节点
- 考虑该节点的颜色和父节点、叔节点的颜色
  - 如果父节点是黑色，无需进行任何操作
  - 如果父节点是红色，叔节点也是红色，只用进行变色操作，具体方法为父节点、叔节点变为黑色，祖父变为红色，递归检查祖父节点。递归终止条件为检查到该节点不存在父节点，或者不存在叔节点，或者不再为两个红色

- 如果是父节点为红色，叔节点为黑色，考虑该节点、父节点、祖父节点的位置关系，如果是在一条线上的只需要一次左旋或者右旋，否则先做一次左/右旋变成一条直线上的，再按照一条线上的旋转。
- 考虑旋转后的颜色，进行变色

红黑树的优点

在有大量的写的操作时比自AVL树要更优

红黑树的缺点

红黑树在查找的时候速度要比AVL树慢