RedBlackTree

红黑树

描述：

红黑树（Red Black Tree）是目前最广泛使用的一种自平衡二叉查找树。AVL树在插入和删除操作中，从根节点向下找到指定节点完成插入或删除操作后，需要再从底向上回到根节点，对途中的所有节点判断是否需要LL、RR、LR、RL四个自平衡操作，从底向上的时间复杂度为，因此AVL树的插入和删除操作的时间复杂度也可以看作，而红黑树在从底向上的过程中只需要2-3次的自平衡操作，不需要遍历数量的节点，因此是目前速度最快的一种自平衡二叉查找树。红黑树的查找、插入、删除操作的平均时间复杂度都是，高度为。

红黑树是2-3-4树的一种等同，很多算法教科书介绍2-3-4树是因为它是理解红黑树算法的重要工具，但2-3-4树本身在实践中并不经常食用。

除了二叉查找树的基本性质，红黑树还具有以下5点性质：

1. 节点是红色或黑色的；
2. 根节点是黑色的；
3. 所有空节点（nil节点）是黑色的；
4. 每个红色节点必然有两个黑色子节点，或者说，任意从根节点向下到叶子结点的路径上不能有两个连续的红色节点；
5. 从任意节点向下到叶子结点的所有路径上都包含相同数量的黑色节点；

如图：



在红黑树的插入和删除操作中，需要对树进行染色和旋转等操作来保证红黑树的上面5条性质。

插入，默认情况下新节点e染色为Red：

1. 若新节点e的插入位置为根节点，没有Father，则e为Black以满足性质2；否则进入情形(2)；
2. 若新节点e的插入位置的父节点Father为Black，不需任何改变就满足性质4；否则进入情形(3)；
3. 若新节点e的插入位置的父节点Father和叔父节点Uncle都是Red，则将Father和Uncle染色为Black，并将祖父节点GrandFather染色为Red，又由于GrandFather可能是根节点，则再对GrandFather进行情形(1)操作；