红黑树（RBT）定义

- 是一个BST（是一个高级的二叉查找树）

  > 二叉搜索树，也称二叉排序树，所有非叶子结点至多拥有两个儿子（Left和Right）；所有结点存储一个关键字；.非叶子结点的左指针指向小于其关键字的子树，右指针指向大于其关键字的子树；

- 每个结点要么是红的，要么是黑的。
- 根节点是黑的，并定义NULL为黑色
- 如果一个子结点是红色，那么它的两个儿子肯定都是黑色，且父节点也必定是黑色。
- 对于任一结点而言，它到叶结点的每一条路径都包含相同数目的黑色节点，称为黑高。

隐含的性质

- 任意一颗以黑色节点为根的子树也必定是一棵红黑树（递归定义）
- 左（右）子树的高度最多是右（左）子树的两倍，即：若 H（left） > H（right），则 H（left）<= 2 * H（right） + 1

时间复杂度

能保证在最坏情况下，基本的动态几何操作的时间均为O（lgn）

红黑树相比于BST和AVL树有什么优点？

红黑树是牺牲了严格的高度平衡的优越条件为代价，它只要求部分地达到平衡要求，降低了对旋转的要求，从而提高了性能。红黑树能够以O(log2 n)的时间复杂度进行搜索、插入、删除操作。此外，由于它的设计，任何不平衡都会在三次旋转之内解决。当然，还有一些更好的，但实现起来更复杂的数据结构能够做到一步旋转之内达到平衡，但红黑树能够给我们一个比较“便宜”的解决方案。

相比于BST，因为红黑树可以能确保树的最长路径不大于两倍的最短路径的长度，所以可以看出它的查找效果是有最低保证的。在最坏的情况下也可以保证O(logN)的，这是要好于二叉查找树的。因为二叉查找树最坏情况可以让查找达到O(N)。

红黑树的算法时间复杂度和AVL相同，但统计性能比AVL树更高，所以在插入和删除中所做的后期维护操作肯定会比红黑树要耗时好多，但是他们的查找效率都是

O(logN)，所以红黑树应用还是高于AVL树的. 实际上插入 AVL 树和红黑树的速度取决于你所插入的数据.如果你的数据分布较好,则比较宜于采用 AVL树(例如随机产生系列数),但是如果你想处理比较杂乱的情况,则红黑树是比较快的

红黑树相对于哈希表，在选择使用的时候有什么依据？

权衡三个因素: 查找速度, 数据量, 内存使用，可扩展性。

总体来说，hash查找速度会比map快，而且查找速度基本和数据量大小无关，属于常数级别;而map的查找速度是log(n)级别。并不一定常数就比log(n) 小，hash还有hash函数的耗时，如果你考虑效率，特别是在元素达到一定数量级时，考虑考虑hash。但若你对内存使用特别严格， 希望程序尽可能少消耗内存，那么慎用hash，特别是当你的hash对象特别多时， hash的构造速度较慢。

红黑树并不适应所有应用树的领域。如果数据基本上是静态的，那么让他们能够插入，并且不影响平衡的地方会具有更好的性能。如果数据完全是静态的，做一个哈希表性能可能会更好一些。

在实际的系统中，例如，需要使用动态规则的防火墙系统，使用红黑树而不是散列表被实践证明具有更好的伸缩性。Linux内核在管理vm_area_struct时就是采用了红黑树来维护内存块的。红黑树通过扩展节点域可以在不改变时间复杂度的情况下得到结点的秩。

面试题

a. 红黑树是一种平衡二叉查找树（简称平衡树，常见的平衡树有AVL，红黑树。AVL树（平衡二叉查找树）的性质为：每个结点的左右子树的高度之差的绝对值最多为1。）
b. 红黑树的各项操作（插入、删除、查找等）复杂度都为log(n)
c. 红黑树的五大特征
1）节点要么为红色，要么为黑色。（不然为啥叫红黑树；））
2）根节点为黑色。
3）叶子节点为黑色。（这两个简直送分，最上面和最下面都是黑的）
4）每个红色节点的左右孩子都是黑色。（保证了从根节点到叶子节点不会出现连续两个红色节点）
5）从任意节点到其每个叶子节点的所有路径，都包含相同数目的黑色节点。（4,5是使得红黑树为平衡树的关键）