# 二叉查找树

左树所有节点值均小于或等于他的根节点值

右树所有节点值均大于或等于它的根节点值

左右子树一定分别为二叉排序树

查找所需的最大次数等同于二叉查找树的高度

# 红黑树

平衡的二叉查找树，最长路径不超过最短路径的2倍

节点是红色或者黑色，根节点为黑色，每个叶子节点都是黑色的空节点，每个红色节点的两个子节点都是黑色， 从任意节点到其每个叶子的所有路径都包含相同的黑色节点。

因为性质5所以默认新插入的节点都为红色

为了符合规则：插入新节点时需要：变色或者旋转(左旋转，右旋转

注意：红黑树的时间复杂度是O(logn)

# AVL树

二叉查找树

空树或者左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是平衡二叉树。

插入和删除：有单旋转（左，右）和双旋转（先左后右，先右后左）

AVL树的查找、插入和删除在平均和最坏情况下都是O(logn)。

红黑树不追求完全平衡，因而在增加或者删除节点时根据不同的情况一般比AVL树旋转次数少；（RB树最多只需要旋转3次实现复衡）

红黑树失衡后的复平衡rebalance的效率更高

AVL高度平衡所以查找效率高

总结：实际应用中，如果搜索次数远远大于插入和删除，那么选择AVL；如果差不多，就RB

# B树B+树

降低二叉树高度，减少磁盘访问。

在B树上查找的过程：顺着指针查找结点--在结点上查找关键字

对B树中序遍历进行顺序查找树中的元素的时候，往返于每个结点之间意味着我们必须得在硬盘页面之间进行多次访问。

B+树：（应文件系统所需）出现在分支结点中的元素会被当做它们在该分支结点位置的中序后继者中再次列出（区别于B树：每个元素在该树中只出现一次）。且每个叶子结点都会保存一个指向后一个叶子结点的指针。

m（结点的最大孩子数）阶B和B+的区别：

* 有n棵子树的结点中包含有n个关键字
* 所有叶子结点包含所有的关键字信息以及指向这些关键字记录的指针，叶子结点本身通过关键字的大小自小而大的顺序链接
* 所有分支结点可以看成是索引，结点中仅包含其子树的最大或者最小关键字。

B+适合带有范围的查找，比如查找学校18-20岁的学生人数，可以从根节点触发找到第一个18岁的学生然后再叶子结点上按顺序查找符合范围的所有记录