Chapter 7 搜索结构

搜索

基于关键码搜索/基于属性搜索

动态搜索环境与静态搜索环境

平均搜索长度(ASL)：搜索过程中关键码的平均比较次数或磁盘平均读写次数，是衡量搜索算法的时间效率的标准

静态搜索结构

静态搜索表：基于数组，下标作为地址

顺序搜索：ASL约为$\frac{n+1}{2}$

有序搜索表

- 顺序搜索：分为成功时的ASL和不成功的ASL
- 折半搜索：时间复杂度可到$O(log_2n)$

二叉搜索树

二叉搜索树：满足以下性质的二叉树：每个结点都有关键码且各结点的关键码不同，左子树所有结点关键码小于根的关键码，右子树所有结点的关键码大于根的关键码，左右子树也是二叉搜索树

二叉搜索树的中序遍历是有序序列。n个结点的二叉搜索树有 $Catalan(n) = \frac{1}{n+1} \times C_{2n}^{n}$ 个

基本操作

- 搜索：递归实现，小于根则向左子树搜索，大于根则向右子树搜索
- 插入：先搜索该元素，若不存在则将其插入搜索停止的位置（NULL）
- 删除：不破坏二叉搜索树的性质
    - 左/右子树为空：用右/左子树代替被删除结点的位置

- 左右子树均不空：寻找左子女中序下最后一个结点或右子女中序下第一个结点代替被删除结点，转换至删除被代替的结点处理
- 左右子女均空：直接删除即可

基本操作的代价均在$O(log_2n)$

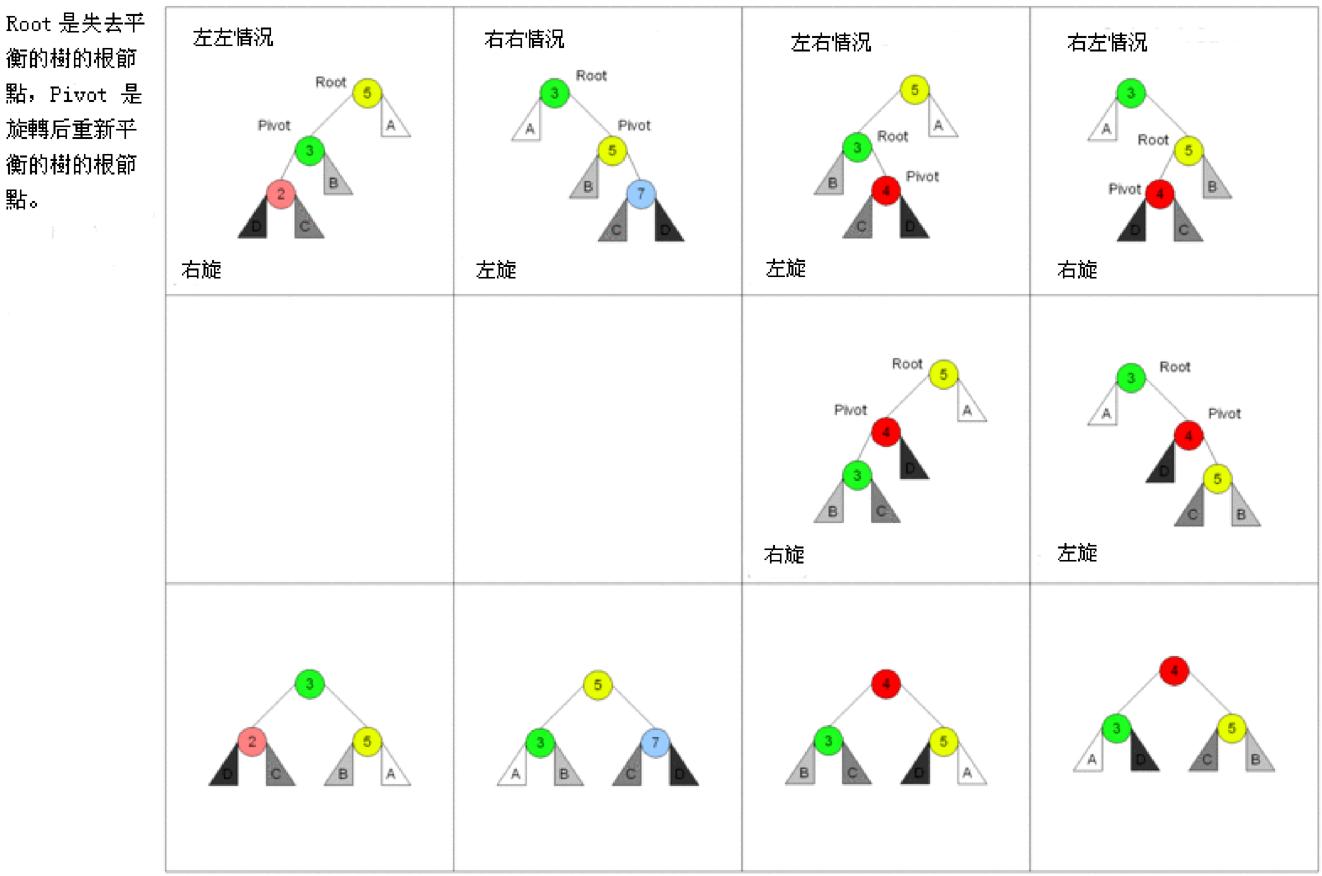
AVL树

AVL树是二叉搜索树，且左右子女的高度差不超过1

旋转

插入结点时保存插入路径，若引起不平衡则从不平衡的结点起沿插入路径向下取两个结点

- 单旋转：三个结点处于一条直线
- 双旋转：三个结点处于一条折线



插入

插入后若某个结点平衡因子的绝对值$|bf| > 1$则出现了不平衡，设新插入的结点为 p，从 p 到根的路径上所有结点的平衡因子都有可能发生变化，因此要向根回溯调整

新结点的平衡因子为0，因此考察其父结点 p_r

1. p_r 的平衡因子为0：结点平衡且高度没有增减，此时结束平衡化，退出程序
2. p_r 的平衡因子绝对值为1：结点没有失去平衡但高度发生变化，需要继续回溯处理其父结点
3. p_r 的平衡因子绝对值为2：失去平衡，需要根据情况做旋转操作恢复平衡，旋转后高度降低，无需回溯

删除

类似二叉搜索树的删除，但不同之处在于删除后要重新平衡化

考察被删除的结点的父结点 p_r

1. p_r 原本的平衡因子为0，删除后变为1或-1，不改变高度，此时可结束平衡化
2. p_r 原本的平衡因子不为0且较高的子树被缩短：此时 p_r 的平衡因子改为0，但高度-1，故仍需回溯平衡化
3. p_r 原本的平衡因子不为0且较矮的子树被缩短：失去平衡，令 p_r 较高的子树为 q
    1. 若 q 的平衡因子为0，则单旋转恢复 p_r 的平衡，平衡后高度没有变化，因此可结束平衡化
    2. 若 q 的平衡因子与 p_r 的平衡因子同号，则单旋转恢复平衡，但高度减少，因此需要继续回溯平衡化
    3. 若 q 的平衡因子和 p_r 的平衡因子异号，则双旋转恢复平衡，但高度减少，因此需要继续回溯平衡化