数据结构知识框架

- 红黑树
  - 概念
    - 红黑树是一棵二叉搜索树，它在每个节点上增加了一个存储位表示该点的颜色，保证最长路径不超过最短路径的两倍，近似平衡
    - 每个结点不是红色就是黑色
  - 性质
    - 根节点是黑色
    - 每个叶子结点（NIL结点，空结点）是黑色的
    - 如果一个结点是红色的，则它的两个子结点是黑色的（没有连续的两个红色结点）
    - 对于每个节点，从该结点到其所包含的叶节点的简单路径上，均包含相同数目的黑色结点（每条路径上黑色结点的数量相同）
    - 每个叶子节点的颜色是黑色（此时叶子结点指的是空节点）
  - 插入实现
    - 初始为空，插入第根结点时先改变颜色
    - 插入结点时父节点为黑色，不违反任何性质，直接插入
      - 情况二
      - 情况四
      - 情况五
  - 删除
    - C++STL用一map/set multimap multiset
  - 适用
    - Java Map
    - linux 内核
    - 其他一些库
  - 红黑树和AVL树的比较
    - 它的左右子树都是AVL树
- AVL树
  - AVL树性质
    - 左子树和右子树高度之差（简称平衡因子）的绝对值不超过 1（-1、0、1）
    - 如果一棵树是高度平衡的，它就是AVL树，如果它的一个子结点，其高度可保持在O(log n)，平均需要 $\frac{1}{2}$log n
  - 平衡化旋转
    - 左单旋
    - 右单旋
    - 左右双旋
    - 右左双旋
  - AVL树的插入
    - 如果查找时，插入结点为根结点，插入直接结点即root
    - 如果找不到，将结点入结点，并在查找过程中将Bfactor，到插入节点直接设即root
    - 更新平衡因子，对树进行调整
    - 递归回溯
  - AVL树的删除
    - 递归回溯结点并检查子树
    - parent同平衡因子为0结点，parent高度不变，则parent的平衡因子及高度不变，不再调整
    - parent同平衡因子非0结点，则parent为新的子树根，若高度变，则需要检查parent的父亲结点的平衡性
    - 递归回溯
    - 结合parent的平衡因子为左情况-2，则需要子结点（即左子结点）的平衡因子
      - 变为删除中结点下的第一个结点
      - 如果parent平衡因子为0，则不需要回溯parent
      - 如果parent平衡因子为parent平衡因子（正负）有别时，则进行一次旋转回溯parent
      - parent同高于树的根为root
      - 如果parent平衡因子为parent平衡因子（正负）有别时，则进行一次旋转回溯parent
- 二叉搜索树
  - 性质
    - 如果左子树不为空，则左子树上所有节点的键小于根结点的键
    - 它的右子树不为空，则右子树上所有节点的键大于根结点的键
    - 它的左右子树分别为二叉搜索树
  - 搜索
    - 根结点小于/大于
    - 根结点Key==要查找的Key，结束查找
    - 根结点Key>查找Key，在其左子树查找
    - 根结点Key<查找Key，在其右子树查找
    - 查找结束/end
  - 插入
    - 查找相应位置插入新的结点
    - 插入时要考虑空树情况，没有重复结点
  - 删除
    - 删除结点
    - 删除结点的子结点有左子节点
    - 删除结点的子结点有右子节点
    - 删除结点的子结点有两个子节点
    - 删除结点的子结点有左右子节点
    - 删除结点的子结点有左右子节点
    - 删除结点的子结点有左右子节点
    - 删除结点的子结点有左右子节点
    - 删除结点的子结点有左右子节点
    - 在它的左子树中找出它的下一个结点（关键结点），并交换结点的键值和数据，并将此结点的子结点的父亲结点的子结点指向它
    - 删除结点的父亲结点
    - 删除结点的子结点
    - 删除结点的父亲结点
    - 删除结点的子结点
    - 删除结点的子结点
    - 删除结点的子