# 平衡二叉树编程

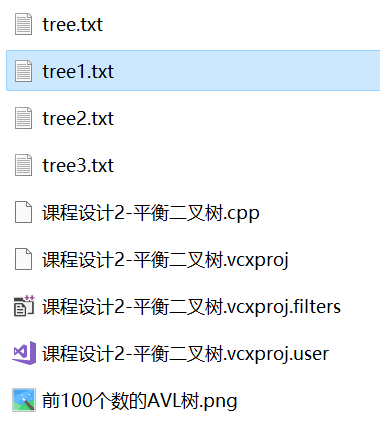
代码行数：325行

算法思想：根据题给数据创建平衡二叉树，依次进行题目中的操作。

主要/核心函数分析：插入：选择合适的叶子结点插入后从该结点开始一直往跟结点开始检查是否平衡，如果不平衡进行平衡化。删除：找到待删除的结点，如果是叶子结点就直接删除再逐层判断是否打破平衡一直到根节点；如果删除结点没有左子树但有右子树，将右子树上移；如果删除结点没有右子树但有左子树，将左子树上移；如果左右子树都存在，如果左子树比右子树高，找到左子树中最大值，将值赋给该结点，取删除左子树最大值这个结点，如果不是，则找右子树的最小值赋给该结点，然后取删除这个最小值结点。查找：结点值比key大，搜索左子树。结点值比key小，搜索右子树。如果搜索到最后结点为空则不存在。

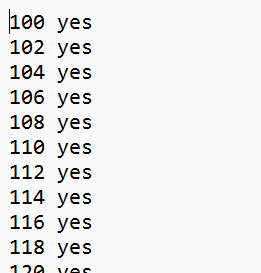
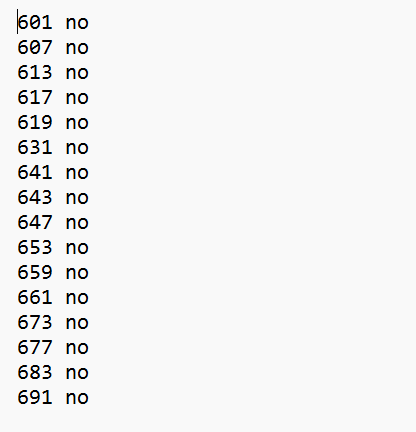
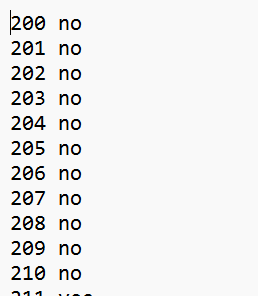
测试数据：对于1-10000的质数序列<2，3，5，7，…，9973>，建立平衡二叉排序树。(1) 依次查询200-300的每个数是否在树中，将结果写入tree1.txt。(2) 依次删除500-2000中的每个质数，再查询 600-700之间的每个质数是否在, 将结果写入文件tree2.txt。(3) 在任务（2）基础上，依次插入1-1000的所有偶数，依次查询100-200的每个偶数是否在树中, 将结果写入文件tree3.txt。

运行结果：



时间复杂度：插入，删除，查询：O（log结点数量）

结果截屏：



依次tree1,2,3.txt部分内容

心得体会：学会了二叉树的插入：插入叶子结点开始逐层平衡化（也就是一个递归的过程）。其中平衡化分为ll，rr，lr，rl这四种。对于删除，学会总结删除的不同情况，然后对树进行调整即可。搜索就是一个一层一层往下的过程，直到搜索不到任何结点。运用递归将树的中序遍历写出来。