# B-树三问

代码行数：465行

算法思想：用tree创建B-tree，书写插入和删除的函数，最后按照题目要求完成task123即可。插入，删除分析在下面。查询则是一直向下搜索直到结点为NULL。

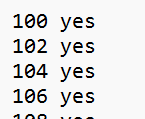
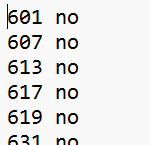
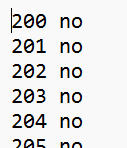
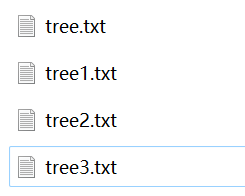
主要/核心函数分析：插入：找到要插入的叶子！结点（也就是最下面），然后如果超出m-1那就取中间进行分裂就好！判断该结点进行调整主要三个方面：1.找出中间结点，把结点前面的键值给新的左孩子，后面的键值给新的右孩子2.将该节点的孩子分给新的左孩子和右孩子3.将中间结点插入该节点的父亲关键字数组中，并将新的左孩子和右孩子插入父亲节点的孩子数组。由最底层逐层向上调整。删除：1.非叶子结点：就选择它左孩子最大值或右孩子最小值进行替代，然后归结为删除叶子结点2.叶子结点就比较复杂了：（1）左兄弟富有或者右兄弟富有（键值个数大于【m/2】-1），就旋转一下（比如：左兄弟富有，将parent该键值给该结点1号键值处，然后将左兄弟最大键值给parent就好了！）（2）都很穷的时候,将parent处键值，左（右）兄弟（键值和孩子），自己合并成一个结点，在插入回parent结点处，别忘了对parent相关值进行改动！和插入一样，需要从底层往跟进行调整，我采用栈，将跟到待删除的叶子结点入栈，然后一个一个取出来（叶子----跟）进行判断是否需要调整。

测试数据：对1-10000的所有质数，建立m=4的B-tree。依次查询200-300的每个数是否在B-tree中，将结果写入文件b-tree1.txt。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b-tree2.txt。然后，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B-tree中, 将结果写入文件b-tree3.txt。

运行结果：见下图

时间复杂度：插入，删除，查询：O（log（结点数））

结果截屏：



结果文件 tree1 tree2 tree3

心得体会：深入理解了Btree的插入和删除。在插入时就是找到要插入的叶子结点然后再进行分裂的过程。和平衡二叉树一样对于分裂需要自己总结书写。删除就比较麻烦，要么叶子直接删，不是叶子还需要找下面的替换再去删除下面的叶子，甚至还有繁琐的合并的过程。对于抽象画树的数据结构有了更深刻的体验，写代码时的注释必不可少！