一、实验选题、实验内容及功能说明

实验选题：查找树的实现与性能分析

实验内容：1. 实现 AVL 树、B 树、红黑树；2. 测试不同数量级节点下各查找树性能；3. 分析并给出理由；4. 建立动态的量化关系。

功能说明：1. AVL树、B树、红黑树的插入，查找，删除，遍历等基本功能；2. 自动化的数据生成器以及自动化时间测试器；3.自动化三维散点图生成器

二、设计方案与设计思路

1. 首先先用C语言实现三棵搜索树的插入，查找，删除等基本功能；

2. 用Python写一个随机数据生成器test.py，将生成的数据输入文本文件data.txt，data1.txt，data2.txt，data3.txt中；

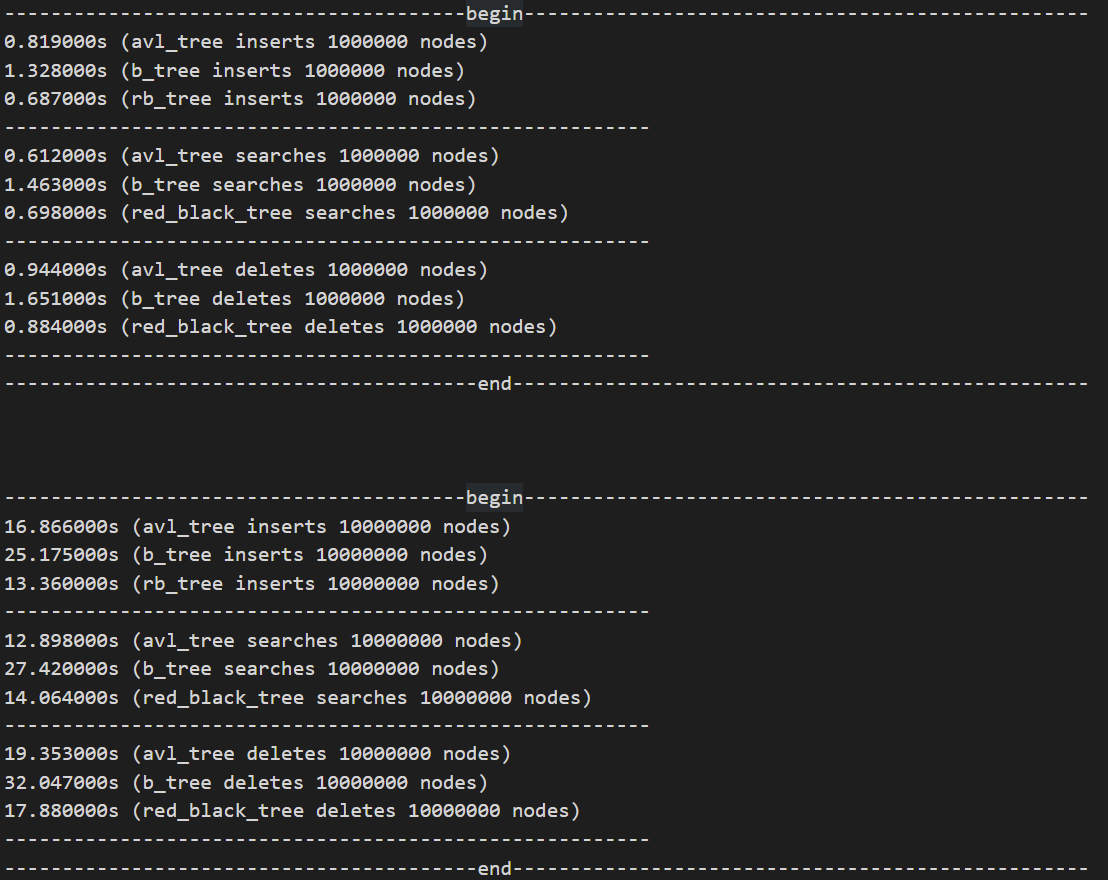
3. 将已经写好的三棵树作为头文件写一个test.c的C程序，将文本文件中的数据读入进行插入，删除，查找3种操作，同时将3棵树对应操作的时间输入到文本文件res.txt中，此时就可以测试不同数量级节点下查找树的性能了；

4. 编写一个控制数据生成器test.py的程序main.py，其中有3个函数，\_test、picture\_2D、picture\_3D其中\_test是测试不同数量级相应操作下3棵树运行时间，picture\_2D是将不同数量级相应操作下3棵树运行时间生成对应的折线图，picture\_3D是将不同比例3种操作下3棵树运行时间生成对应的散点图，当main.py中的函数调用test.py生成好了数据以后，自动运行3中C程序编译后的结果，如果是\_test函数那么结果就保存在了res.txt中，如果是picture\_2D或者picture\_3D函数那么得到res.txt同时读取res.txt中的结果到列表中，然后生成对应的三维散点图或折线图，于是就建立好了动态的量化关系的图，只要分析即可（以下分析中B树均为3阶B树）。

三、程序运行效果

给出程序各种情况下运行结果的截图，并**配以文字说明**。爬虫选题如使用除Python自带库以外的第三方库，也请在此介绍。

1. **测试不同量级节点下各查找树性能：**



上图是res.txt中输出的内容，是先用自动化程序main.py中的\_test调用test.py同时生成10，100，1000，10000，……10000000数量级的插入删除以及搜索操作的数据，再由main.py运行编译好的test.exe就可以了。

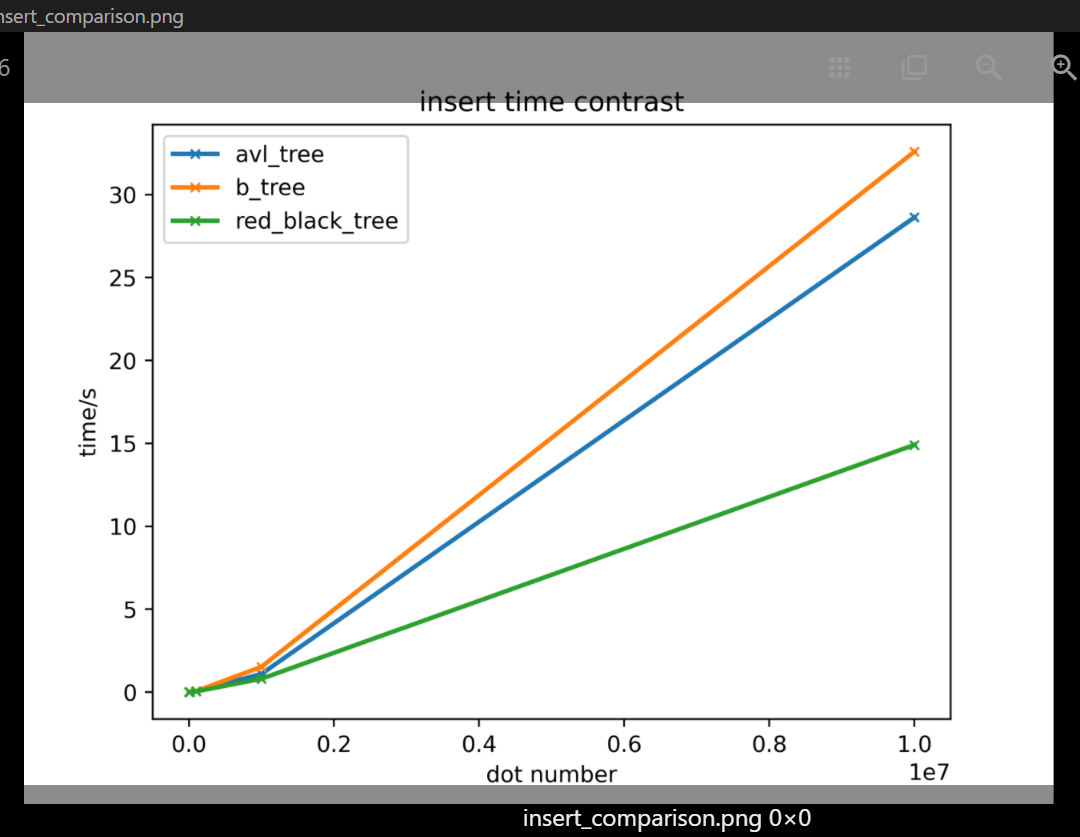
经过运行main.py（10，100，1000，10000，……10000000数量级）共3次得到如res\_1.txt中的结果，由于C语言运行速度较快，于是我不考虑10000以下的数量级产生的结果，分析平均情况可知：

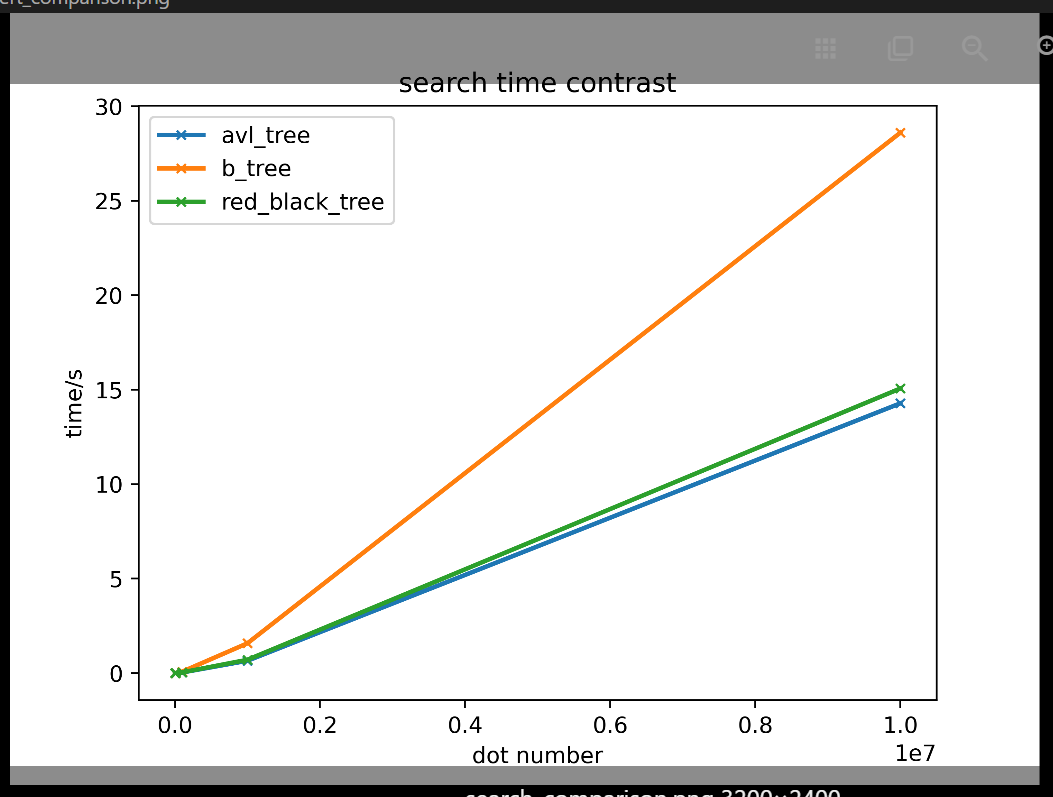
对于插入：红黑树最快，然后是AVL树，最后是B树，对于AVL树由于其每次插入都要维持绝对平衡，向上调整以及旋转次数更多，而红黑树只是维持近似平衡调整过程较少，并且有些情况可以通过染色解决，步骤更加简略，于是红黑树会快于AVL树，至于B树可能是由于其中遍历时使用太多的for循环，同时搜索时没有用二分导致速度下降，第一遍用Python写的时候B树介于红黑树与AVL之间，应该是直接调用了Python的库函数find()，sort()的缘故。

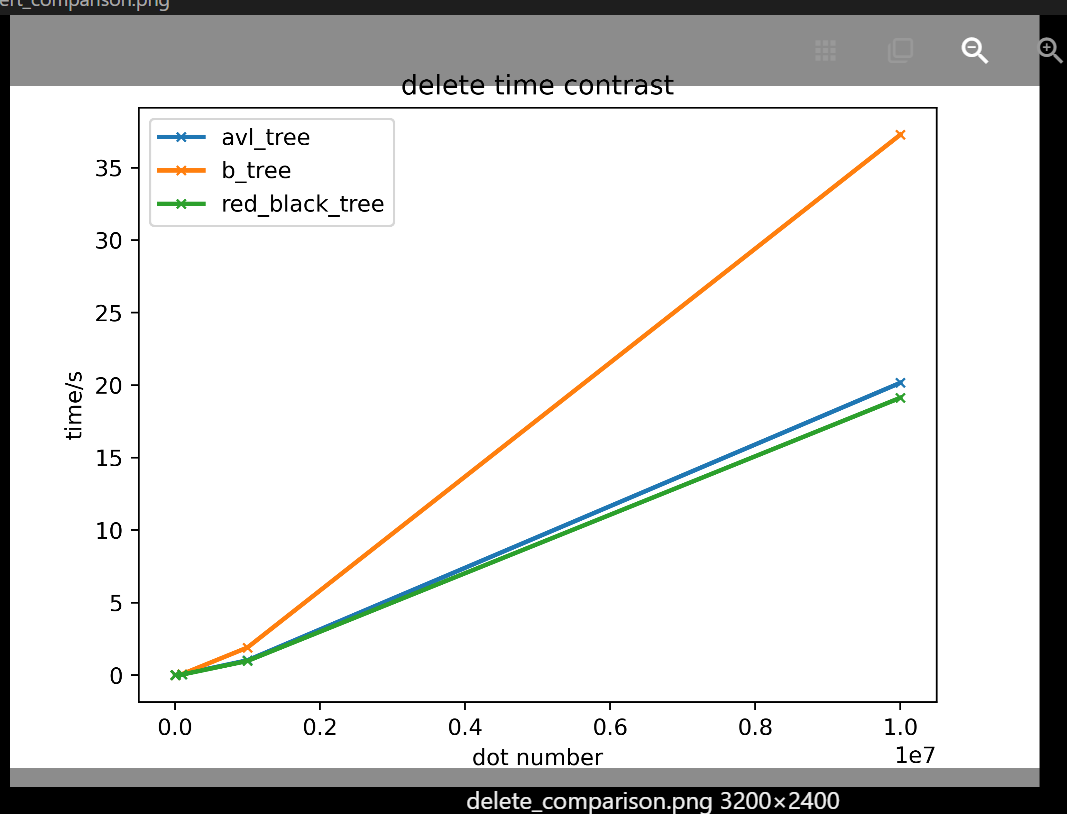
对于搜索：红黑树的搜索略微慢于AVL树，B树还是同样慢的十分异常，理由应该同上，至于红黑树与AVL树同样是因为红黑树只是近似平衡的原因导致红黑树的深度大概率大于AVL树，因此查找起来更慢。

对于删除：红黑树最快，然后是AVL树，最后是B树，B树还是运行过慢，对于AVL树由于其删除向上调整步骤更多，调整步骤均为旋转比染色更加复杂，从而才可以维持绝对的平衡，而红黑树只是维持近似平衡调整过程较少，并且有些情况可以通过染色解决，步骤更加简略，于是红黑树会快于AVL树，至于B树原因应该还是上面说的。

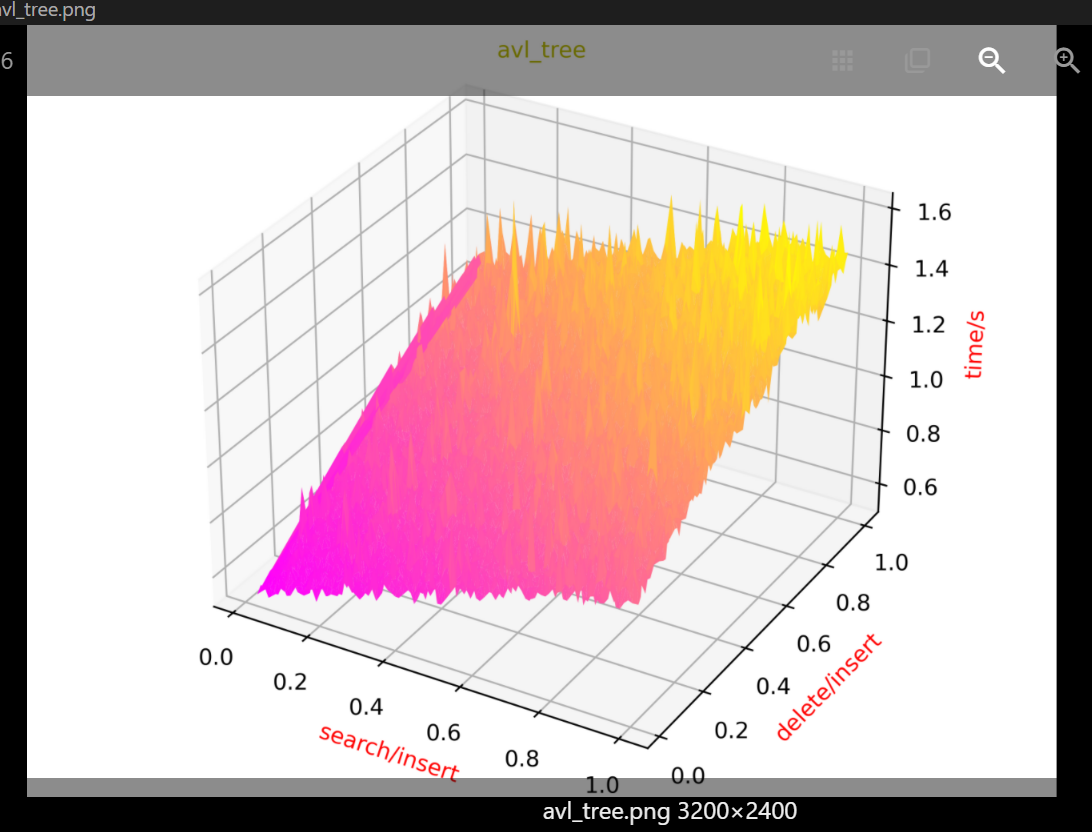
最后附上调用main.py中的picture\_2D生成的折线图，这三幅图是三棵树分别在1000、10000……1000000数量级相应操作的对比，与上述分析以及上述数据基本吻合。

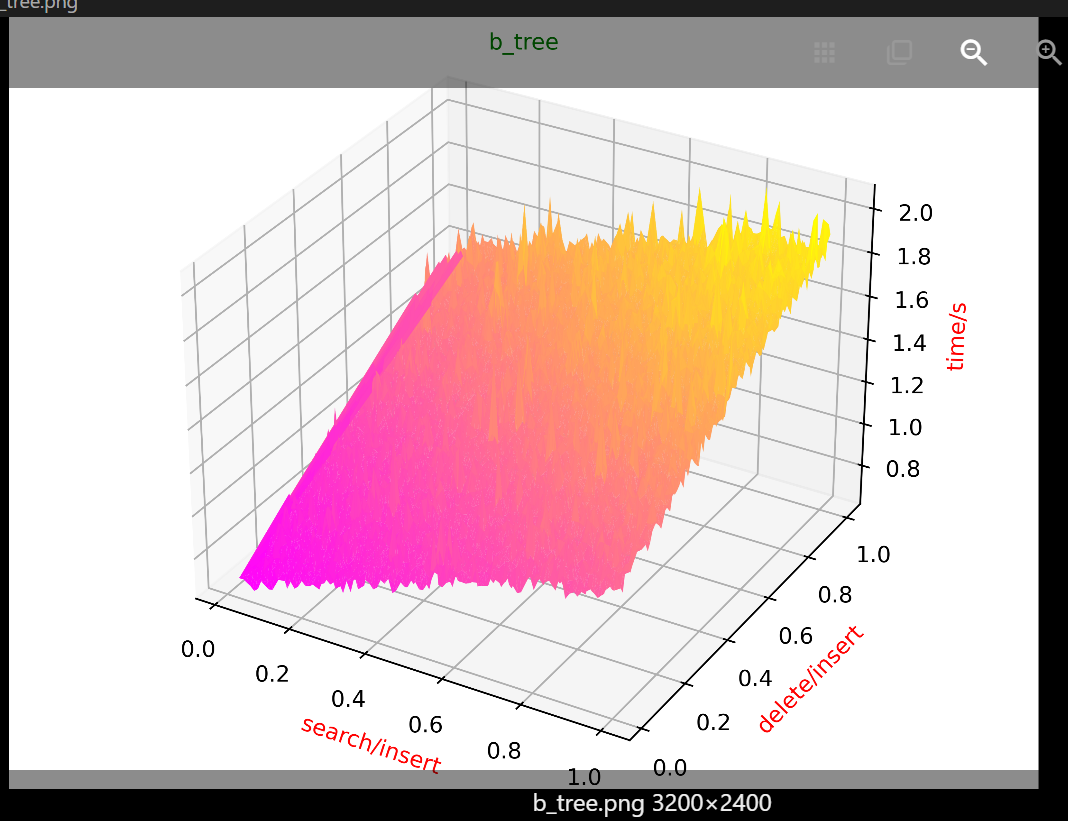


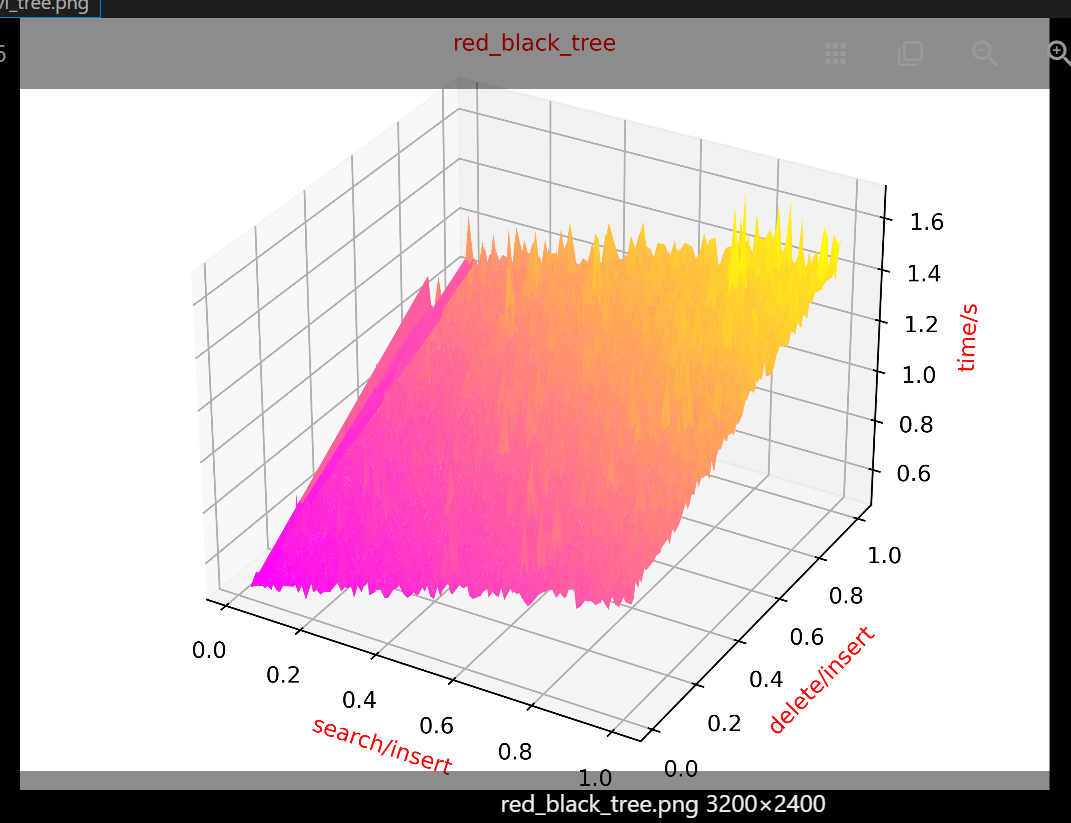




1. **建立动态的量化关系：**

****





这就是得到的散点图，首先是通过main.py中的picture\_3D函数按照删除/插入，查找/插入的不同比例（固定插入为50万，一共一万组数据）生成数据，然后同上得到res.txt的结果，最后将结果读入到列表中然后生成如上所示的散点图。同时由上图可知在数据量足够大时它与其它两棵树的差距其实在缩小，关于散点图的简要分析如下：

由**测试不同量级节点下各查找树性能**中的分析知只有AVL的插入比红黑树快，但是快的也微乎其微，同时由于实际情况中不可能只搜索，不查找与删除，而这两项十分耗时间占主要开销，故在几乎所有情况下用红黑树是效果最好的（由上图也同样可看出）。

最后，在生成散点图的过程中使用了Python的第三方库matplotlib、numpy，用于生成散点图。同时由于数据过于庞大，该处实现是在舍友租的1核2G的服务器中CentOS 8环境下跑出来的结果

四、设计亮点**（选填）**

1. 可以自动的生成散点图，为了更加直观的观察不同操作比例下三棵查找树的性能，便于分析；

2. 可以自动生成不同数量级的同一操作的三棵树的折线图，更清晰的查看不同量级节点下各查找树各操作性能，以及进行比对。

五、实验总结

（1）在实验中遇到了哪些问题？是如何解决的？

1.首先是红黑树，B树的删除操作，刚开始用Python写的一直都有问题，于是重构用C写，当然删除还是有bug，于是在舍友帮助下以及肉眼观察法花了快一天的情况下解决了问题；

2. 然后是如何测试的问题，尝试了好几种测试方案：1. 完全用Python写数据生成以及计时；2. 用Python写数据生成，用C计时……在几种方案都尝试过之后最终选择了第二种方法，主要是因为Python随机数可以生成不重复的数据同时删除时可以随机取插入中的节点，而C做不到，于是跑的就很慢；

3. 最后在用舍友服务器跑散点图时，遇到了内存开销过大的问题跑到一半的进程直接被系统杀死了，于是又花了一个下午发现原来是我跑完一组数据后没有将内存释放的原因，同时我写的B树节点中定义的key，child数组内存也没有释放。

（2）请简要地总结一下自己在数据结构课程中各方面的收获。

1. 了解了基本的栈、队列……等数据结构以及一些基本的诸如KMP、Prim……等初等算法及其实现方式；

2. 了解了如何写简易爬虫从网上爬取目的信息；

3. 通过大作业约1000多行的代码量极大的提高了自己的代码能力，以及debug能力。

（3）你对课程的教学、实验等环节有没有自己的建议？（比如课程知识点安排、实验题目难易等等）

1. 建议教学环节可以重点讲一下代码的实现或者希望老师可以稍微演示一下实现过程；

2. 实验课建议每次布置一定量的课后作业，同时提高题目的难度，比如可以结合多种数据结构以及一些简单的算法进行出题，而不是单一知识点的题目，同时希望增加图论这部分的题目练习，这学期好像就2道图论的上机题目。