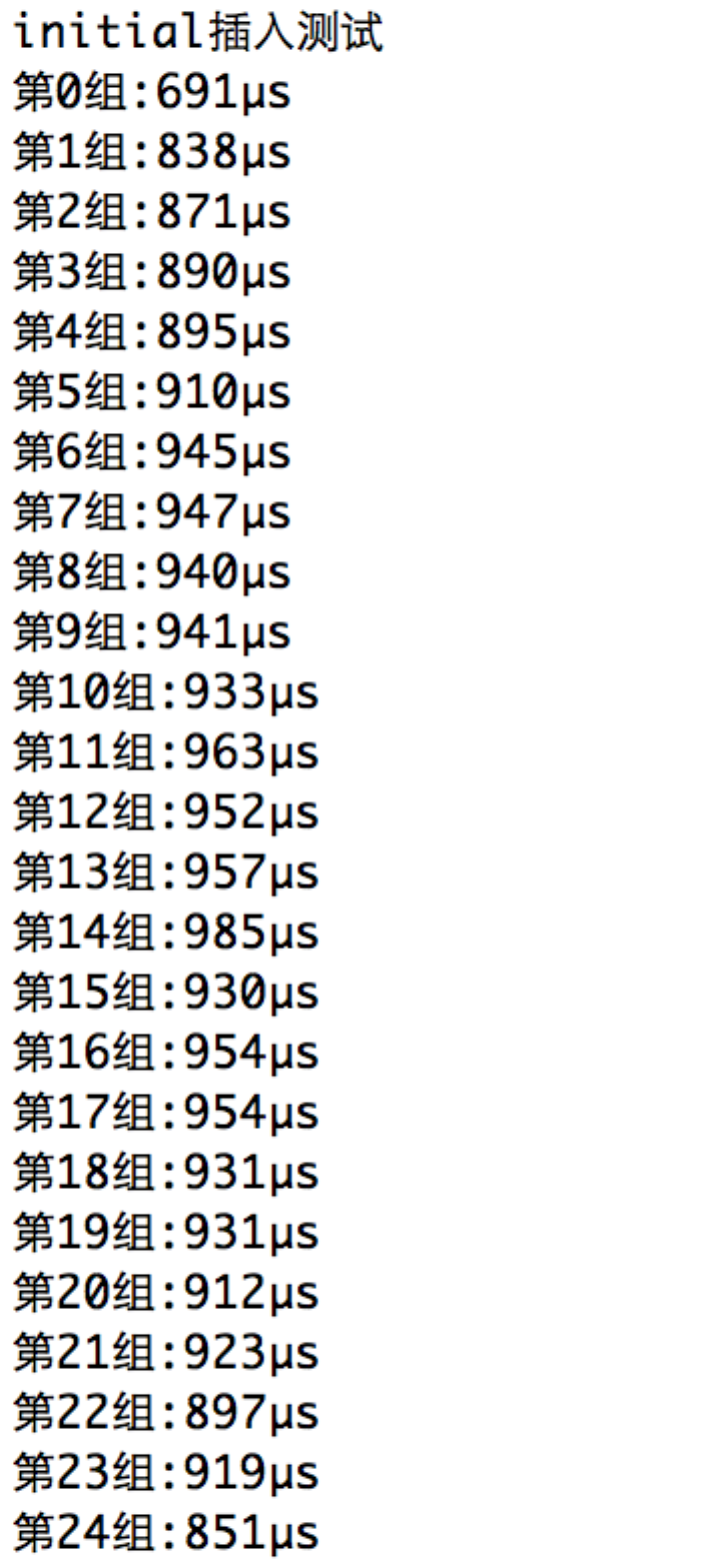
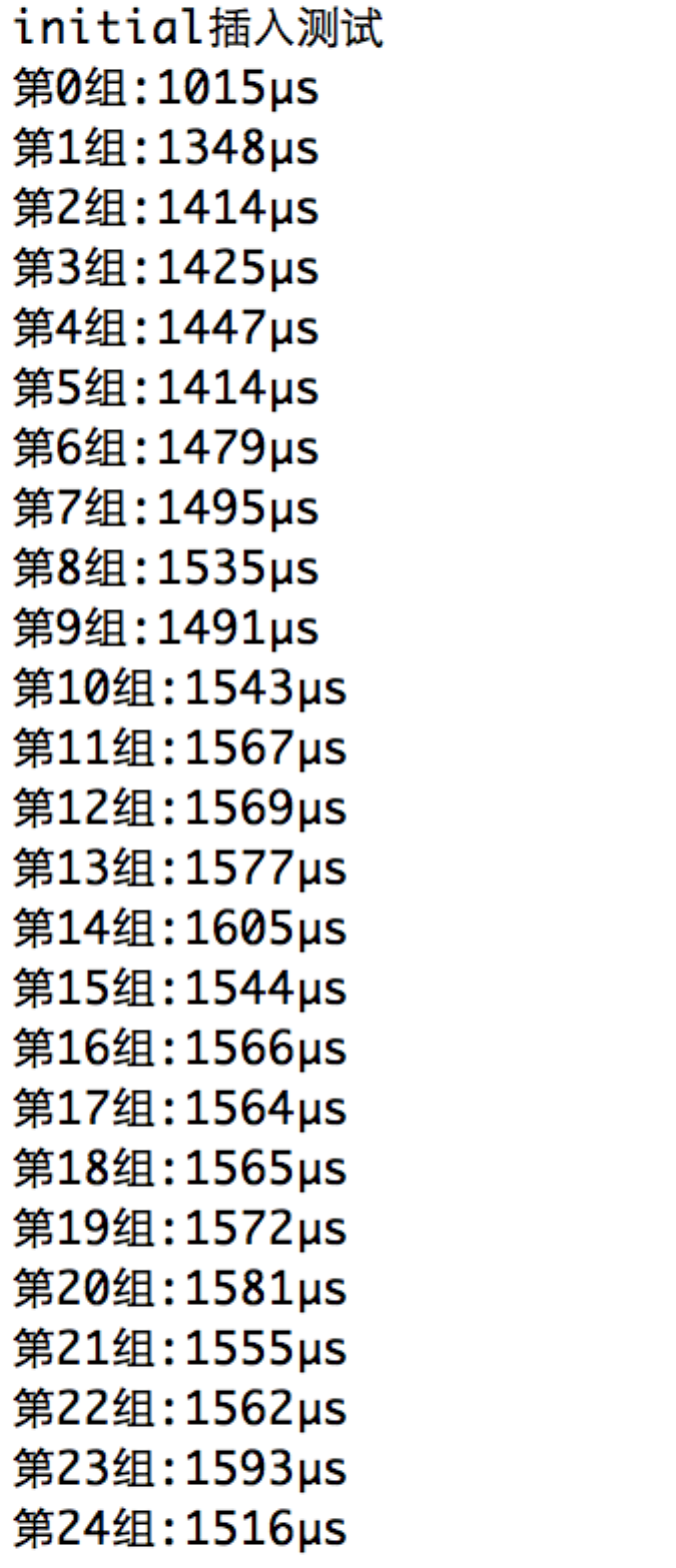
PJ测试文档

本次PJ共写了8个类，每个类的说明及注释都较为详细的写在文件中了。

测试结果分析：

首先Test0是测试红黑树，Test1是测试B树。每个测试共5组。

Test0和Test1中initial插入的测试结果比较（左边是Test0，右边是Test1）

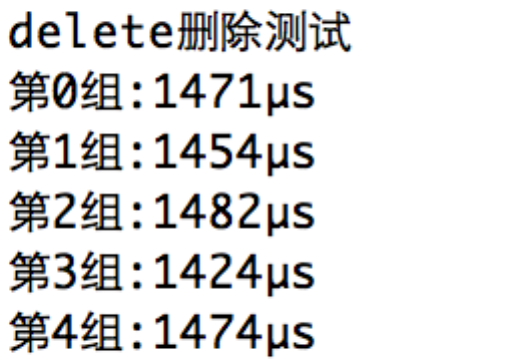
 

共25组，每组是200个单词的时间。测试1000次最后取平均。单位是微秒（μs）

可以看到，红黑树25组的平均值是914.4μs.B树25组的平均值是1501.7μs.

红黑树的5000词插入比B树有着更好的性能，大约提升了39%的性能。

Test0和Test1中delete删除的测试结果比较（左边是Test0，右边是Test1）

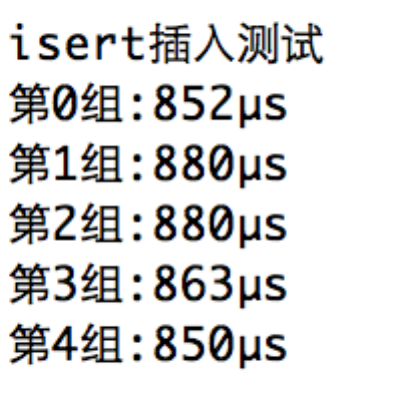
 

共5组，每组是200个单词的时间。测试1000次最后取平均。单位是微秒（μs）

可以看到，红黑树5组的平均值是625μs.B树5组的平均值是1461μs.

红黑树的1000词删除比B树性能上更好。大约提升了57%的性能。

Test0和Test1中insert插入（先initial）的测试结果比较（左边是Test0，右边是Test1）

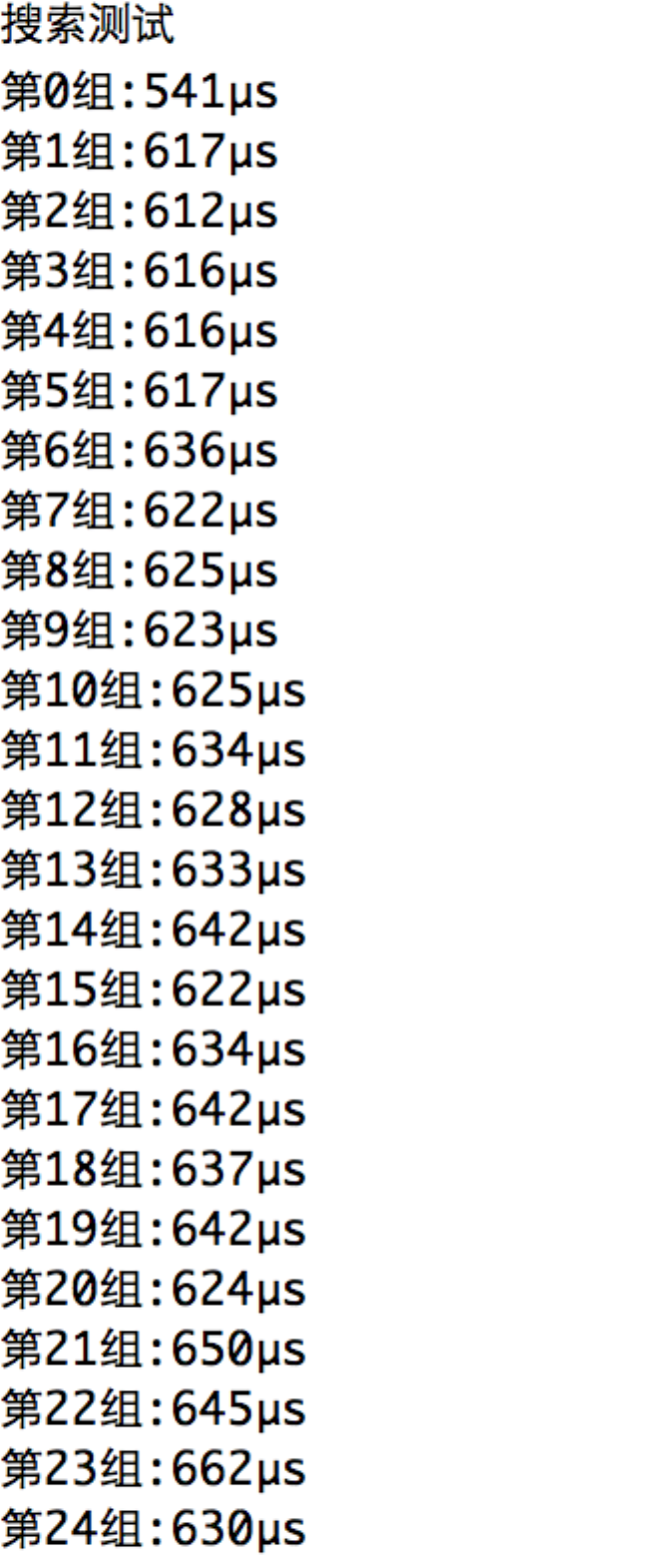
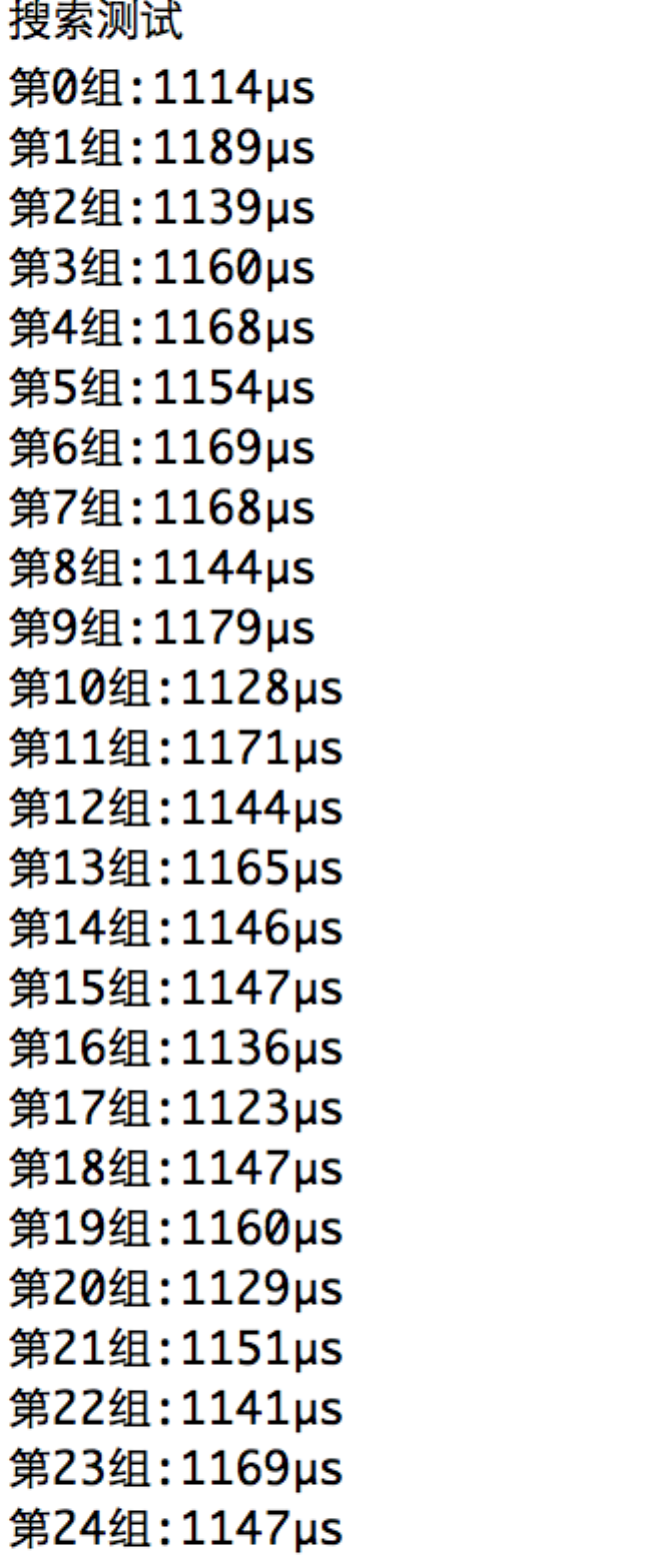
 

共5组，每组是200个单词的时间。测试1000次最后取平均。单位是微秒（μs）

可以看到，红黑树5组的平均值是865μs.B树5组的平均值是1628.4μs.

红黑树的1000词插入比B树有着更好的性能。大约提升了46%的性能。

Test0和Test1中search查找（单个单词，使用1\_initial.txt）的测试结果比较（左边是Test0，右边是Test1）

共25组，每组是200个单词的时间。测试1000次最后取平均。单位是微秒（μs）

可以看到，红黑树25组的平均值是652μs.B树25组的平均值是1151.84μs.

红黑树的5000词查找比B树有着更好的性能。大约提升了43%的性能。

Test0和Test1中rangeSearch查找（测试g-m）的测试结果比较

Test0: 7627

Test1: 7936

重复测试10次最后取平均。单位是微秒（μs）

可以看到，红黑树的范围查找与B树相比性能相差不多。并没有明显的差距。但是红黑树还是略微优于B树。

查找：红黑树和B树的查找代价都是O（lgn），具体来说，红黑树是，而B树是，理论上来说，当t的值越大，B树查找的代价会越小。但本次实验中t的取值为10，与，所以红黑树的查找性能比B树更好一点。

删除和插入：对于本次实验，由于单词量太少，并不会牵扯到磁盘的读写，仅靠内存完全可以存储所有的单词，所以B树在降低I/O操作数方面的优点并没有被体现。同时也是由于单词量少，在红黑树的一次删除或者插入的操作中，修复时进行的旋转要更少，但是对于B树，在预处理和修复时都需要节点中孩子和关键字大量的调整，性能上会比红黑树差很多，事实上，实验也证明了这个问题。

范围查找：模仿了遍历的做法，两者经过的关键字是一样多的，B树需要对节点内部的数组进行操作，红黑树需要调用方法获得孩子节点，两者性能相差不大。但是红黑树稍微比B树要快一点。

综上，红黑树在5000词的大小上无论是插入还是删除性能要比B树更好，范围查找两者相差不多。但是当数据量增大时，B树应当优于红黑树，原因在于B树的树高增长很慢，而红黑树很快，导致当插入或者删除一个数据时，红黑树要反复调用方法获得左右孩子，但是B树只需要在节点内部比较即可。