

## 第七次实验报告

517021910499 陈鸣阳

```
D:\PROGRAM_CPP\517021910499_7_AVL\bin\Debug\517021910499_7_
the time of create avltree(txt1):267.66ms
the time of create avltree(txt2):261.68ms
the time of create bintree(txt1):271.46ms
the time of create bintree(txt2):270.88ms
the time of find avltree(txt1):266.55ms
the time of find avltree(txt2):262.1ms
the time of find bintree(txt1):272.9ms
the time of find bintree(txt2):270.7ms
Process returned 0 (0x0)    execution time : 75.295 s
Press any key to continue.
```

用两个 txt 文件的数据，分别进行创建和查找操作，可以看到无论是创建还是查找，AVL 树用时均比普通二叉查找树要短。

### 创建：

对于查找树的创建而言，理论上来说，普通二叉查找树的创建时间应该短于 AVL 树，因为 AVL 树需要对结点的平衡度进行调整，而普通二叉查找树并不需要。但是测试结果显示 AVL 树的创建时间也短于普通二叉查找树，推测可能和数据的分布有关，或者和电脑本身有关，毕竟样本只有两个，完全可能得出与理论相背的结果。

### 查找：

查找操作的实验结果与理论相符。普通二叉查找树会在一定程度上退化为一个单链表，在最坏情况下，查找时间将是  $O(N)$ 。而 AVL 树对平衡度的调整保证了  $O(\log N)$  的复杂度，故查找所用的时间短于普通二叉树。

```
the time of delete avltree(txt1,10000):0.0504s
the time of delete avltree(txt1,20000):0.1041s
the time of delete avltree(txt1,30000):0.159s
the time of delete avltree(txt1,40000):0.2161s
the time of delete avltree(txt1,50000):0.271s
the time of delete bintree(txt1,10000):50.3ms
the time of delete bintree(txt1,20000):105.6ms
the time of delete bintree(txt1,30000):159.5ms
the time of delete bintree(txt1,40000):218.75ms
the time of delete bintree(txt1,50000):276.75ms
Process returned 0 (0x0)    execution time : 24.496 s
Press any key to continue.
```

### 删除:

由于两个 txt 文件数据相近, 只需对一个 txt 文件的情况进行考察。

由所得到的数据可以看出, 在树的规模较小时, 两者的时间差别很小, 随着树的规模的增大, 两者的时间差别也开始变大。这也是与理论相符合的。树的规模越大, 普通二叉查找树越容易向单链表退化, 退化后所需的时间成本也会上升。而 AVL 树虽然花了一定的时间对平衡度进行调整, 但节省的时间更多, 是合算的。

综上, 在树的规模较小时普通二叉查找树和 AVL 树的查找(推测可知)和删除操作时间差别不大, 但是 AVL 树的创建有额外的对平衡度进行调整的时间(理论如此)。所以我认为树的规模较小时可考虑采用普通二叉查找树。当树的规模较大时, 普通二叉查找树的退化所带来的时间成本的上升就变得比较明显, 此时可考虑采用 AVL 树。