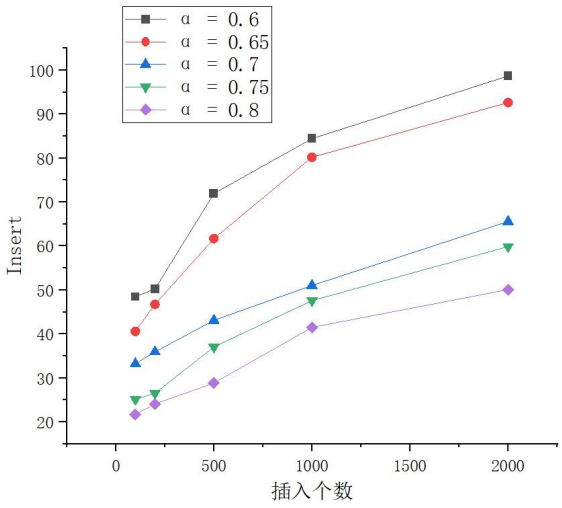
1. α 对基本操作性能的影响：

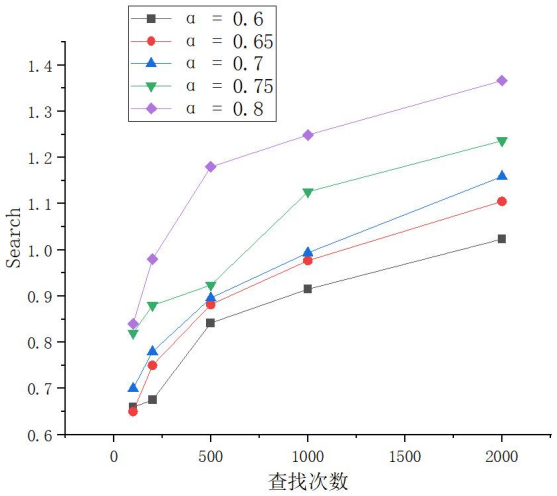
1、insert：

Insert（微秒）	α =0.6	α =0.65	α =0.7	α =0.75	α =0.8
100	48.47	40.6	33.22	25.17	21.71
200	50.245	46.74	35.927	26.535	24.095
500	71.952	61.708	43.114	37.058	28.864
1000	84.419	80.193	51.033	47.596	41.503
2000	98.655	92.631	65.5685	59.8775	50.0925



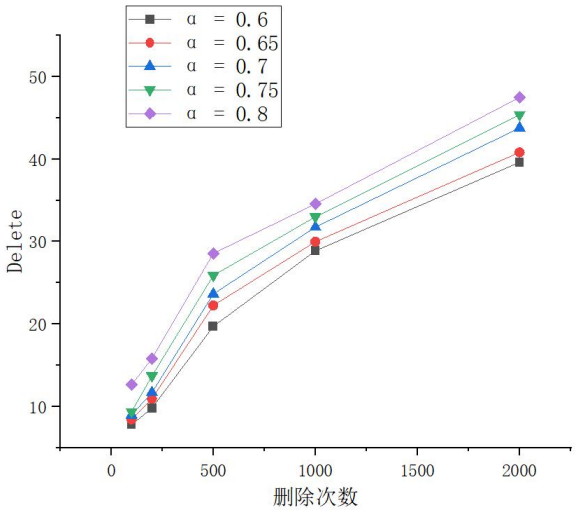
2、search 操作：

Search（微秒）	α =0.6	α =0.65	α =0.7	α =0.75	α =0.8
100	0.66	0.65	0.7	0.82	0.84
200	0.675	0.75	0.78	0.88	0.98
500	0.842	0.882	0.896	0.924	1.18
1000	0.915	0.977	0.994	1.126	1.249
2000	1.0235	1.105	1.159	1.2365	1.3665



3、delete 操作：

Delete（微秒）	α =0.6	α =0.65	α =0.7	α =0.75	α =0.8
100	7.79	8.44	8.97	9.37	12.65
200	9.81	10.94	11.7	13.72	15.82
500	19.716	22.26	23.654	25.896	28.576
1000	28.888	29.99	31.757	32.993	34.571
2000	39.622	40.821	43.775	45.395	47.4885



实验结论：从上图可以看出，随着 α 取值的增长，insert 的性能逐渐提升，而 search 与 delete 的性能均出现明显下降。这与理论值是相符的。并且，从总体而言，所有曲线都是满足对数时间复杂度的。但是仍然有部分曲线的数据点出现了一定的偏差，这可能是由于测试次数较少(每个数据是 5 测量之后取平均数值的结果)等原因导致的。

2. 与基于旋转实现的树的性能比较：（本次实验选择的用于比较性能的树为 AVL 树）

通过对两种树的性能进行测试之后得出如下结果：

Insert（微秒）	ScapegoatTree	AVLTree
100	39.81	6.48
200	41.925	6.965
500	58.93	10.012
1000	70.806	11.791
2000	139.449	23.585

Search（微秒）	ScapegoatTree	AVLTree
100	0.5	0.48
200	0.57	0.56
500	0.56	0.564
1000	0.593	0.595
2000	0.81	0.7665

Delete（微秒）	ScapegoatTree	AVLTree
100	9.57	7.12
200	17.475	15.645
500	28.852	27.828
1000	76.562	75.235
2000	183.601	182.066

通过上图可以得出如下结论：在测试范围内可以发现，AVL 树的插入性能是远远优于替罪羊

树的插入性能的，对于查找而言，两者性能相近；对于删除操作而言，两者性能相近，但是一般情况下为 AVL 树的删除性能要优于替罪羊树的。

综合上述结果，给出如下建议：

1. 在需要进行大量插入操作时，相比于替罪羊树，选择利用旋转进行平衡的二叉平衡树可能效果更好；
2. 对于大量的查找操作，选择替罪羊树或者利用旋转进行平衡的平衡树都可以，两者性能相近；
3. 对于删除操作，两者性能相近，但是选择类似于 AVL 树这样的利用旋转进行平衡的二叉平衡树可能效果更好。