ScapegoatTree实验报告

实验数据

实验一.ScapegoatTree性能与Alpha以及树的大小的关系

在Alpha、数据规模不同的情况下，我对ScapegoatTree进行了大量的测试，实验数据如图，表中数据为执行对应指令的平均时间。

Alpha\Size	1000	2000	5000	10000	20000
0.60	0.0095	0.0215	0.0591	0.0877	0.1757
0.65	0.0100	0.0218	0.0555	0.0888	0.1867
0.70	0.0118	0.0255	0.0520	0.0903	0.1825
0.75	0.0091	00200	0.0553	0.0931	0.1850
0.80	0.0092	0.0216	0.052	0.0927	0.1832

表1.ScapeGoatTree的插入平均用时与Alpha和Size的关系（**单位（ms)**）

Alpha\Size	1000	2000	5000	10000	20000
0.60	0.0575	0.0792	0.0899	0.0836	0.0980
0.65	0.0605	0.0764	0.0879	0.0857	0.0972
0.70	0.0747	0.0955	0.0936	0.0850	0.0976
0.75	0.0583	0.0792	0.0954	0.0863	0.0980
0.80	0.0591	0.0726	0.0900	0.0847	0.0990

表2.ScapeGoatTree的查找平均用时与Alpha和Size的关系（**单位（μs)**）

Alpha\Size	1000	2000	5000	10000	20000
0.60	2.304	4.102	8.855	15.68	33.63
0.65	3.210	7.210	15.28	24.73	47.57
0.70	3.747	8.634	19.19	32.27	60.67
0.75	5.250	12.19	27.885	46.54	88.19
0.80	7.650	20.20	37.60	65.71	124.9

表3.ScapeGoatTree的删除平均用时与Alpha和Size的关系（**单位（μs)**）

实验二.ScapegoatTree与AVLTree在插入、查找、删除方面的性能比较

插入

tree\Size	1000	2000	5000	10000	20000
AVL树	0.3845	0.8099	2.4346	4.7654	9.9926
替罪羊树	9.6112	35.7454	270.131	1196.54	4048.91

查找

tree\Size	1000	2000	5000	10000	20000
AVL树	0.0569	0.1279	0.4021	0.9719	2.3411
替罪羊树	0.057	0.1282	0.447	1.0498	1.9468

删除

tree\Size	1000	2000	5000	10000	20000
AVL树	0.0176	0.0351	0.0877	0.1823	0.4321
替罪羊树	6.9813	29.7032	193.018	655.527	2504.49

实验结论

实验一

现象：

- 可以看到，在插入、删除、查找的过程中，随着树节点数目的增多，每次指令需要的时间逐渐增加，且符合log(n)的时间复杂度
- Alpha逐渐增大时，插入的用时变化不大，查找总体上用时逐渐增大，删除有明显增大

结论：

- 由此可知，ScapegoatTree的插入、查找、删除总体上是时间复杂度为log（n）的操作；
- 理论分析，alpha增大会导致插入时进行的重构减少，用时应当减少，但是实验数据上并没有体现出来，原因可能和插入的数据情况有关，因每次插入都是随机生成的数据，可能会与较大的时间出入；
- 另一方面，查找和删除与理论分析吻合地较好，验证了alpha大小对scapegoatTree的影响
- 综上可知，如果在一个工程中需要大量使用删除操作，可以适当减小alpha

实验二

现象：

- 在各种数据规模上，AVL树在插入和删除上的性能均显著优于Scapegoat树
- 二者在查找速度上相近

结论：

- 从实验数据来看，无法得出Scapegoat树相较于AVL树的优点
- 从实现来看，Scapegoat树不涉及旋转操作，思路简单
- 另一方面，Scapegoat树具有可调节的Alpha值，可以针对不同的问题采取不同的应对方式