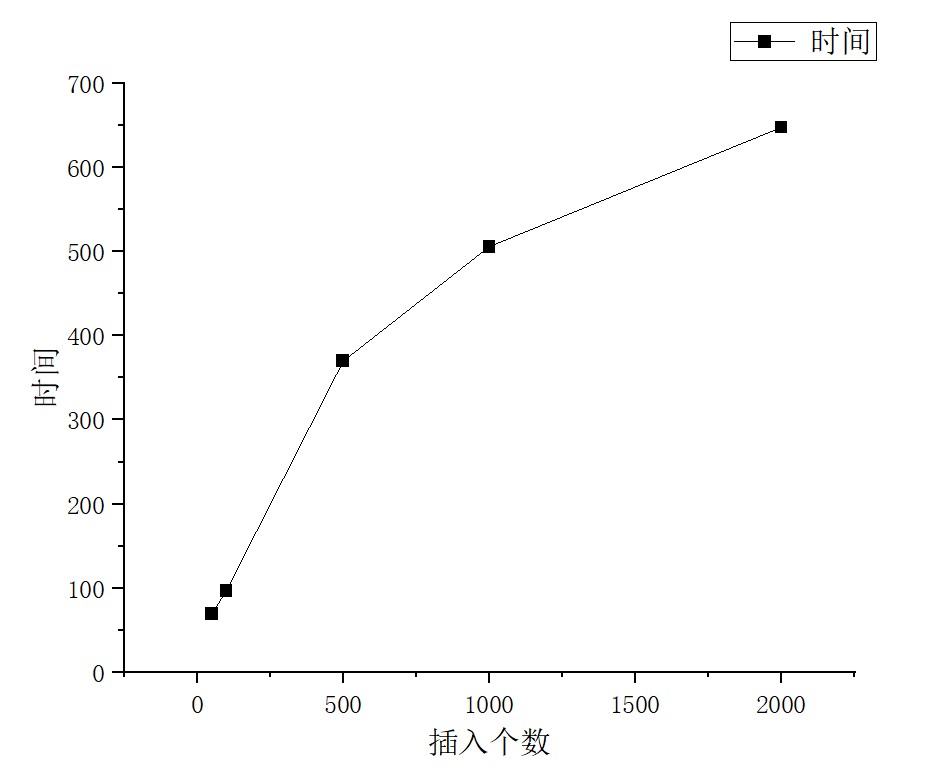
任务1：

对于不同数量的插入进行时间的测试：



从上图可以知道，AVL树在插入过程中确实具有O(logn)的时间复杂度。但是，从上图曲线可以看到，当插入为2000个时，测试数据相对于log曲线会出现一定的偏移，并且在小于500的部分其实曲线趋于线性，这可能是由于使用了递归方式进行插入导致的时间偏长。同时，实验次数过少而出现了一定的偶然误差也会对结果产生一定的影响。

任务2：

|  |  |  |
| --- | --- | --- |
|  | AVL树 | 跳表 |
| 顺序插入 | 15142 | 198110 |
| 随机插入 | 10705 | 32295 |
| 倒序插入 | 12952 | 55056 |

从上图可以看出，对于插入操作，AVL树的性能好于跳表。

任务3：

对结果的分析：

1.对于任务1：

曲线总体形状是符合log曲线形状的。

对于理论数值，其每次插入走过的平均路径为1.5\*logn个节点，这说明随着插入个数的增加，插入总时间确实是应该满足logn时间复杂度的，从总体而言，任务1的结果较好。

2.对于任务2：

AVL树每次的插入平均需要走过1.5\*logn个节点，而对于跳表其每个元素都是有序排列的，股每次插入的平均时间应为O(N)，故跳表插入的时间性能应该弱于AVL树。这点任务2的结果已经有所体现。

单看AVL树的结果，发现其随机插入的时间最小，且倒叙插入的时间小于顺序插入的时间。

这可能是因为随机插入相对于定序插入更能体现平均的时间性能，故时间最短。而对于倒序以及顺序插入两种，个人认为两者的时间应该相近，但是结果出现了一定的误差，这可能是受到了递归算法的性能的影响。同时，作为一个需要收集数据的实验，该结果也可能受到实验次数较少等情况的影响。

1. AVL树与跳表的使用场景：

根据上述结果可知，AVL树相对于跳表插入性能较优秀。个人认为AVL树更适用于需要进行大量的插入操作的场景，而跳表可能更加适合于插入操作较少（尤其是定序插入），但是对查找的需求量很大的场合。