****

**算法分析与设计预习报告**

****

**学 院： 电子信息与人工智能学院**

**专业名称： 计算机科学与技术**

**班 级： 计算机203**

**学 号： 202007020625**

**姓 名： 於俊涛**

**任课老师： 齐 勇**

**实验2 查找算法性能分析实验**

班级 计算机203 学号 202007020625 姓名 於俊涛 角色reader

**一、实验目的**

1、掌握7大查找算法的原理。

2、分析不同算法的性能及适用环境。

**二、实验环境**

线上平台sustoj.com。

**三、实验内容**

**（一）单一算法时间复杂度比较**

****

**（二）多种算法性能比较**

已知各个算法的原理及时间复杂度，在什么情况下使用这些算法，才能将效率达到最高呢？

**1、折半查找与插入查找比较**

折半查找的前提条件是需要有序表顺序存储，对于静态查找表，一次排序后不再变化，折半查找能得到不错的效率。但对于需要频繁执行插入或删除操作的数据集来说，维护有序的排序会带来不小的工作量，那就不建议使用。

插值查找：对于表长较大，而关键字分布又比较均匀的查找表来说，插值查找算法的平均性能比折半查找要好的多。反之，数组中如果分布非常不均匀，那么插值查找未必是很合适的选择。

**2、散列查找与二叉查找树比较**

哈希表有以下几个突出的缺点：

当更多的数插入时，哈希表冲突的可能性就更大。对于冲突，哈希表通常有两种解决方案：第一种是线性探索，相当于在冲突的槽后建立一个单链表，这种情况下，插入和查找以及删除操作消耗的时间会达到O(n)，且该哈希表需要更多的空间进行储存。第二种方法是开放寻址，他不需要更多的空间，但是在最坏的情况下（例如所有输入数据都被map到了一个index上）的时间复杂度也会达到O(n)。

所以，在决定建立哈希表之前，最好可以估计输入的数据的size。否则，resize哈希表的过程将会是一个非常消耗时间的过程。例如，如果现在你的哈希表的长度是100，但是现在有第101个数要插入。这时，不仅哈希表的长度可能要扩展到150，且扩展之后所有的数都需要重新rehash。

哈希表中的元素是没有被排序的。然而，有些情况下，我们希望储存的数据是有序的。

另一方面，我们讨论二叉树：

二叉树不会有冲突（collision），这意味着我们能够保证二叉树的插入和查找操作一直都是O(log(n))的时间复杂度。

二叉树的空间占用跟输入的输入数据一致。所以我们不需要为二叉树预先分配固定的空间。所以，你也不需要预先知道输入数据的size。

所有的元素在树中是排序好的。

如果你预先知道输入数据的大小，而且有足够的空间储存哈希表，且不需要对数据进行排序，那么哈希表总是好的。因为哈希表在插入，查找和删除操作中只需要常数时间。

另一方面，如果数据是持续的加入，你预先不知道数据的大小，那么二叉树是一个折中的选择。

**（三）时间复杂度分析**

1、并查集

在read函数中，时间复杂度为O（n），并查集的时间复杂度为O（n2）。

2、二叉查找树

二叉查找树时间复杂度为O（n2）。

3、中序遍历

中序遍历时间复杂度为：O（n2）。