# 数据结构与算法基础课程实验报告

# 实验 3：查找结构的实验比较

<table>
<tr><td>姓名</td><td>李大鑫</td><td>院系</td><td colspan="2">计算机学院</td><td>学号</td><td colspan="2">1170300825</td></tr>
<tr><td>任课教师</td><td colspan="3">臧天仪</td><td>指导教师</td><td colspan="3">臧天仪</td></tr>
<tr><td>实验地点</td><td colspan="3">格物</td><td>实验时间</td><td colspan="3"></td></tr>
<tr><td rowspan="2">实验课表现</td><td colspan="2">出勤、表现得分 10%</td><td></td><td rowspan="2">实验报告<br>得分 40%</td><td rowspan="2"></td><td rowspan="2">实验总分</td><td rowspan="2"></td></tr>
<tr><td colspan="2">操作结果得分 50%</td><td></td></tr>
</table>

实验目的：

1. 掌握二叉搜索树的实现
2. 掌握二叉搜索树中的添加、删除、查找操作
3. 掌握二分查找的实现
4. 比较二叉搜索树与二分查找进行查找的时间性能

实验内容：

本实验要求编写程序实现 BST 存储结构的建立（插入）、删除、查找和排序算法；实现折半查找算法；比较 BST 查找结构与折半查找的时间性能。

实验要求：（学生对预习要求的回答）（5 分）　　得分：

1、 设计 BST 的左右链存储结构，并实现 BST 插入（建立）、删除、查找和排序算法。

BST 的建立通过插入实现，BST 的操作中都需要用到 BST 遍历的基础知识，就是通过将 key 与节点进行比较然后确定是在左子树还是右子树。

删除操作需要额外的注意，因为删除之后还需要维护二叉搜索树的基本性质。

2、 实现折半查找：

算法简单实现没有难度。

3、 实验比较？

实验比较的比较标准是平均查找长度，所谓平均查找长度就是 N 次查找中平均一次查找需要进行比较的次数，用这个标准来衡量一个查找进行的快与慢的问题。

实验过程中遇到的问题如何解决的？（10 分）（着重从软件调试、质量保证、结果分析方面进行阐述）　　得分：

问题 1：

如果进行性能比较？首先构造 BST 的时候区分两种情况，一种是按照已经排好顺序的数字来建树，一种是按照经过一定随机之后的数字进行建树，首先随机的实现是在已经排好序的数组基础上进行交换得来的，对每一个数组元素进行交换，交换的对象是通过 random() 得到的，这样就实现了一个基本随机的序列。其次进行性能比较的时候对两种查找情况进行分别计算，一种是能够查找得到，一种是不能查找到，分别计算这两种情况下的平均查找长度。进行建树的数字选择 0-2048 之间的所有奇数，查找成功的情况依次使用这些奇数，查找不成功的情况使用这些范围内的偶数。这样实现了数据的平均分布，计算出来的平均查找长度是有可信度的。

问题 2：

在二叉搜索树中实现删除的思路：如果我们需要删除一个节点，我们先查找它在不在这棵树中，如果不在报错，如果在的话，我们在树中找到这个节点，删除这个节点之后我们依然需要维护二叉搜索树的基本性质，这里需要进行讨论：如果只有一棵子树，则直接将子树提上来，否则代表两棵子树都存在，这时候我们选择右子树中值最小的节点，将它作为这颗子树新的根节点，因为它是右子树中的最小节点，所以比右子树所有节点都要小，比左子树中所有节点都要大，所以这棵子树依然符合二叉搜索树的定义。对于这个过程使用一个 deleteMin 函数来递归查找。

| 本次实验的体会（结论）（10 分） | 得分： |
| --- | --- |

本次主要用实现了 BST 和二叉查找，然后通过比较这两种算法的查找得出了平均查找长度作为性能的衡量。

1） 对于 BST 的删除操作，个人感觉这种维护树的基本性质的算法十分值得借鉴。

2） 对于比较的结果：显然按照顺序建立的 BST 性能十分差，其查找次数与原值在建树序列中的位置成正比，平均查找次数（两种情况）为 512。随机建树的性能显然更好，平均查找次数能查找到的情况平均为 11.55 不能查找到的情况平均为 12.55，对于二分查找，能查找到的情况平均为 9，不能查找到的情况平均为 10，相比之下二分查找性能更好一些，因为我们随机得到的序列显然不是完全随机的，因此建树的时候 BST 不能保证完全平衡，这就使一次查找的深度（比较次数）会偏离 O(logn)，而达不到二分查找 O(logn)的效果。

3） 是否说明：就平均性能而言，BST 的查找与折半查找差不多，为什么？个人认为如果在数据完全随机进行建树的条件下，BST 的查找与折半查找差不多，但是如果是根据有规律的数字进行建树的话就会导致 BST 的性能下降。

| 思考题： | |
| --- | --- |
| 思考题 1： | 得分： |

思考题 1：查找的方法都有哪些，特点是什么？

1、 线性查找

用线性表示数据集合时，通过遍历线性集合，对每一个元素进行比较来查找目的数据。

特点：实现简单，但是效率低。

2、 折半查找

在已经实现元素按照关键字从小到大排序的序列中查找，1）初态：令 low=0，表示查找范围的下界；up=last，表示查找范围的上界；2）折半：令 mid=(low+up)/2，取查找范围中间位置上元素的下标。 3）如果以 mid 为下标的元素是查找元素则返回，否则如果查找元素更小，则 up=mid-1，否则 low=mid+1。直到不满足 low<=mid 则停止。

特点：折半查找方法找到元素的比较次数最多不应超过相应二叉判定树的高度。所以折半查找法的时间复杂性是 O(logn)。

3、 分块查找

这种方法要求将表中的元素均匀分为若干块，每一块中的元素可以是任意排列的，而各块之间要按一定的顺序排列，块中的元素满足块之间的顺序要求。再建立一个线性表用来存放每一块中最大或者最小的关键字，把这个线性表成为索引表。首先在索引表中进行二分查找，查找到元素对应的块，然后，在块中线性查找元素是否存在。

特性：分块查找拥有顺序查找和二分查找的双重优势，即顺序查找不需要有序，二分查找的速度快。

| 思考题 2： | 得分： |
| --- | --- |

思考题 2：课本上介绍了哪几种搜索树，简单说明他们的特点？

1、二叉搜索树

特点：实现简单，但是时间性能与建树时候的顺序有关。

2、AVL 树

特点：平衡二叉树，其平均和最坏情况下的查找、删除、插入的时间都是 O(logn)。他或者是一棵空二叉树，或者是具有以下性质的二叉查找树：其左子树和右子树都是高度平衡的二叉树，且左子树和右子树的高度之差的绝对值不超过 1。

3、B-和 B+树

特点：

B-树是一棵的 m-路的查找树，它或者为空，或者满足以下性质：

1） 根节点至少包含 2 个儿子

2） 除根节点和失败节点以外，所有节点都至少具有 m/2 个儿子

3） 所有失败节点都在一层

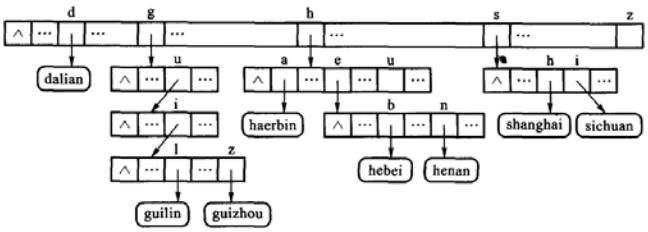
B+树与 B-树的差异在于：

1） 有 k 个子节点的节点必然有 k 个关键码

2） 非叶节点仅具有索引作用，与记录有关的信息均存放在叶节点中

4、 Trie 树

特点：Trie 树包含两种类型的节点：元素节点和分支节点。元素节点存放整个关键字以及其他相关信息，分支节点存放子树的指针和在分支节点结束的关键字的元素节点的指针。在 Trie 树的每一层根据关键字的一个字符来确定分支，将关键字分为 27 个互不相交的类。如下图：




指导教师特殊评语：

指导教师签字：

日期：