哈尔滨工业大学

数据科学与大数据技术专业

实验报告

课程名称：计算机算法基础

实验项目：查找结构的实验比较

实验题目：BST 查找结构与折半查找方法的实现与实验比较

实验日期：2020.12

班级：1804105

学号：1180400510

姓名：石瑞河

|  |  |  |
| --- | --- | --- |
| 设计成绩 | 报告成绩 | 指导老师 |
|  |  | 张岩 |

**一、实验目的**

1.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；

2. 实现折半查找算法

3. 比较 BST 查找结构与折半查找的时间性能

**二、实验要求及实验环境**

**实验要求：**

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。

2．实现折半查找算法。

3．实验比较：设计并产生实验测试数据，考察比较两种查找方法的时间性能， 并与理论结果进行比较。

**实验环境：**

硬件环境



软件环境

VSCode Mingw-w64

**三、设计的基本思想和算法的原理**

BST（二分查找树）：

BST有如下性质：左子树上的所有结点的值都小于根结点，右子树上的所有结点值都大于根节点，其左子树和右子树也是BST。

根据BST的性质，可以很好的实现数据的动态增删查改

查询：从根结点开始，与当前结点key值比较，相等则返回，key小于节点值则去左子树查找，大于节点值则去右子树找。若当前结点为空则表示无此key值

增加：从根节点开始，添加的key值与当前结点值比较，小于则添加到左子树中，大于则添加到右子树中，当前结点为空，则写入添加的key值和信息。

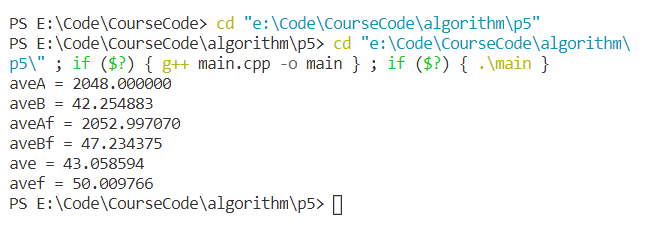
删除：删除不存在结点，不操作；删除叶结点，将父节点保存的此结点指针清空，并释放内存；删除非叶结点，找到左子树的最大值结点或者右子树的最小值结点A，用来代替当前结点，让后断开清空A的父节点保存的A的指针。

改：删除，再添加

二分查找：

二分查找是在已经排序好了的序列上，通过查找值和中间值的比肩，每次将查找范围缩小一半的查找方法，是一种静态的查找方法。

**四、实验结果与分析**



树A为顺序插入建立的一颗BST，树B为随机顺序插入的一颗BST

aveA表示树A成功查找到某值的平均时间（count次数）

aveA = 2048, 平均查找一个值需要2048次计算或比较

aveB表示树B成功查找到某值的平均时间（count次数）

aveB = 42.25, 平均查找一个值只需要42.25次计算或比较

aveAf表示树A查找某值失败的平均时间（count次数）

aveAf = 2052.997, 树A查找一个不存在的值平均要计数2053.00次

aveBf表示树B查找某值失败的平均时间（count次数）

aveBf = 47.23, 树B查找一个不存在的值平均要计数47.23次

使用二分查找，查找成功平均计数42.05次，查找失败平均计数50.00次

和树B的查找性能相差不大

BST的查找性能和树的形态关联较大，树A是顺序插入建立的，实际上已经退化成了一条链，树的深度很深，查找性能差；树B堆数据打乱后再插入建树，树的形态比较好，比较平均，深度小，查找性能就比较好，平均下来单次查找为O(logn)，常数和二分查找也相近。

二分查找每次查找性能基本一致，平均下来单次查找O(logn)。

就平均性能，还不能认为BST查找与折半查找性能差不多，BST在最优形态下与二分查找平均性能相似，但是在建树过程中，很难保证BST的形态如何，因此BST查找的平均性能也不稳定，在O(logn)到O(n)之间;而二分查找的平均性能是稳定的O(logn)

如果要保持BST的功能，并不断调整形态来保持性能，就需要使用splay，AVL-tree或者RB-tree等数据结构，在建树过程中调整形态，查找性能保持平均的O(logn)

**五、经验体会与不足**

BST的实现较为简单，没有调整形态的操作，造成了BST的性能受影响，不稳定。

**六、附录：源代码（带注释）**

见附件：BST.cpp binary.cpp main.cpp