

**实验报告**



**题目： 实验二：树-查找综合应用实验**

**班 级： 2023211321**

**学 号： 2023212872**

**姓 名： 计子毅**

**学 院： 计算机学院**

**2024年 12 月 3 日**

1. 实验目的
2. 练习二叉树的实现及操作，学习用二叉树解决实际问题；
3. 练习查找树的实现及操作，学习用查找树解决实际问题；
4. 锻炼递归算法编写及程序调试的能力；
5. 学习自己查找相关资料以解决实际问题的能力。

二、实验环境

操作系统：windows10

语言环境：c++

编译器：Visual Studio

三、实验内容

**场景设定：**

假设某超市需要实现一个商品库存管理系统。每种商品有唯一的ID，初始的10k种商品样例数据如附件csv所示。文件第一行为表头，后续每行为一种商品，包含Uniq Id（唯一id）,Product Name（商品名称）,Inventory（库存）三列。因为有大量的查询、添加、删除操作，计划采用查找树作为其核心数据结构。请在此场景设定下完成以下要求。

**要求：**

1. 读取marketing\_sample\_10k\_data.csv文件，依次插入10k项商品，构造出对应的二叉查找树。
2. 求出该二叉查找树的高度。
3. 求针对这10k项商品的平均查找长度。
4. 采用非递归的方式中序遍历该二叉树，依次输出商品的Uniq Id。
5. 读取drop sample\_1k\_data.csv中的1000项需要删除的商品id清单，依次从二叉查找树中将其删除。
6. 求删除后的二叉查找树的高度。

**附加要求**：将要求第1步中构造二叉查找树，优化成构造平衡二叉树。不要求实现平衡二叉树结点的删除。

四、实验步骤（基础80分+附加10分）

1.文件数据处理

引入头文件fstream，在函数inputfile中打开文件marketing\_sample\_10k\_data.csv，将文件内容去掉表头后的数据构建结构体数组products，其结构为id name inventory。在函数inputfile1中打开文件drop sample\_1k\_data.csv，去掉表头后，将数据存入数组Dp中。当两个文件被成功打开后会输出successful。

矩形

描述已自动生成

2.构建二叉查找树

首先编写节点信息，创建结构体BSTnode，包含string data；BSTnode\* left；BSTnode\* right。

创建树根节点变量，并将其初始化，即将根节点设为空。

编写二叉查找树的插入函数InsertBST，如果需要插入的树为空，则将数据插入到根节点处，并将左右子树设为空。根据二叉查找树的性质，若待插入数据小于根节点数据则递归到左子树进行插入，若大于根节点数据则递归到右子树进行插入。

编写构建二叉查找树函数CreateBST，使用循环，通过InsertBST，将10000个数据插入树中。

3.求二叉树高度

编写求长度函数int Height(BST& T1)，如果传入的树是空树，则返回高度为0，分别用两个变量存放左右子树的递归求高度结果，返回1+左右子树高度的最大值。



4.求二叉查找树的平均查找长度

将平均查找长度分为查找失败和查找成功两种情况，分别由ASL\_fail和ASL\_successful求出。

1. ASL\_fail：如果传入的树是空树，则证明满足查找失败的情况，此时返回该节点的深度weight，若不是则递归返回左右子树的平均查找长度之和，并将深度加一。
2. ASL\_successful：如果传入的树是空树，查找失败则返回0。若非空，则递归返回目前该点的深度weight加上左右子树的平均查找长度（将weight加一）。

蓝色的标志

描述已自动生成

5.采用非递归方法中序遍历二叉查找树

编写中序遍历函数Midorder，编写栈数据结构以及与其相关的初始化，插入，取出，判空操作，并传入函数中。当栈和传入的树补同时为空时，证明该树并没有遍历完毕，则利用while循环判断这一条件。再设置一层while循环当树不为空时，将树的根节点压入栈中，并将树指针移向左子树根节点，直到将左子树全部压入栈中，即树为空时，跳出第二层循环。随后从栈中弹出节点，并访问它的数据，并将树指针移向该节点的右子树，重复第一层循环。

文本

描述已自动生成

…

…

图片包含 室内, 照片

描述已自动生成

7.二叉查找树的删除并求出高度

编写删除函数Delete，返回值为BST。若传入的树是空树，则返回这棵树；若需要删除的数据小于传入的树的根节点的数据，则使树的左子树等于在左子树递归删除该数据之后的返回的树；若需要删除的数据大于传入的树的根节点的数据，则使树的右子树等于在右子树递归删除该数据之后的返回的树；若数据等于根节点的数据，则需要分为以下四种情况：

1. T为叶子节点，此时只需要删除该叶子节点即可，则将该节点变为空，并返回该节点
2. T只有左子树，则需要将该节点删除，并将左子树变为父节点的子树，即将该节点指针移向左子树根节点，在返回该节点指针。
3. T只有右子树，则需要将该节点删除，并将右子树变为父节点的子树，即将该节点指针移向右子树根节点，在返回该节点指针
4. T具有左子树和右子树，此时需要找到T的直接前驱节点替代T，并将前驱结点删除，即T的左子树的最大值节点。

编写总删除函数DeleteBST，通过循环将1000个待删除的数据通过Delete函数从树中删除，最后通过前面的函数求出高度及平均查找长度

文本

中度可信度描述已自动生成

7.附加要求：构建平衡二叉树

首先我们需要先将节点按照二叉查找树的插入方式插入树中，并同时判断传入的待插入树是否因插入变得不平衡，这样可以保证实时记录最小不平衡子树。在判断因插入而失去平衡后，我们分为一下四种情况进行树的修改：

1. 在左子树的左子树插入：编写LL函数，以最小不平衡子树的根节点为中心进行单次右旋，即将插入节点的父节点变为根节点，根节点变为父节点的右子树，并将父节点的右子树变为根节点的左子树。
2. 在右子树的右子树插入：编写RR函数，以最小不平衡子树的根节点为中心进行单次左旋，即将插入节点的父节点变为根节点，根节点变为父节点的左子树，并将父节点的左子树变为根节点的右子树。
3. 在左子树的右子树上插入，编写函数LR，此时涉及根节点，爷节点，父节点的旋转，将父节点单次左旋，再单次右旋，最终到达根节点，旋转过程与单次旋转相同。
4. 在右子树的左子树上插入，编写函数RL，此时涉及根节点，爷节点，父节点的旋转，将父节点单次右旋，再单次左旋，最终到达根节点，旋转过程与单次旋转相同

最后编写函数BuildAVL，通过循环将10000数据插入平衡二叉树中。

文本

描述已自动生成

五、实验分析和总结（10分）

本实验设计树的数据结构以及查找的思想，总体来说难度较大，要求实验者具备扎实的知识基础。主要难点在于二叉查找树的删除，需要考虑较多可能性；以及平衡二叉树的构建，其中对于不同情况的旋转函数的编写较为不易。其余部分可通过简单的递归完成要求。

不成功的设计：在非递归中序遍历中使用到栈的数据结构，但对于栈数据结构的记忆已经模糊，使得在编写链栈时采用了顺序栈的代码，使得在中序遍历时出现空间混乱的情况出现。

注意：在删除后求高度，若使用左子树最大节点替换树的高度会变成31；而采用右子树最小节点替换树的高度仍为32。

六、程序源代码（10分）

见附件。

其它说明：总分超过100分的计为100分。