二叉排序树

#### 软件工程20级2班 2053300 胡锦晖

# 项目概述

## 项目背景

### 二叉排序树的基本定义

二叉排序树又称二叉搜索树、二叉查找树，是基于节点的二叉树树数据结构，具有以下属性：

结点的左子树只包含小于当前结点的数。

结点的右子树只包含大于当前结点的数。

左右子树也必须是二叉排序树。

不能出现重复的结点。

不难看出二叉排序树是通过递归定义的，所以后续关于二叉排序树的一些基本算法也将通过递归实现。

### 1.1.2 二叉排序树的优势

构造一棵二叉排序树的目的，并不是为了排序，而是为了提高查找和插入删除关键字的速度。不管怎么说，在一个有序数据集上的查找，速度总是要快于无序的数据集的，而二叉排序树这种非线性的结构，也有利于插入和删除的实现。

二叉排序树是以链接的方式存储，保持了链接存储结构在执行插入或删除操作时不用移动元素的优点，只要找到合造的插入和删除位置后，仅需修改链接指针即可。插入删除的时间性能比较好。而对于二叉排序树的查找，走的就是从根结点到要查找的结点的路径，其比较次数等于给定值的结点在二叉排序树的层数。 极端情况，最少为 1 次，即根结点就是要找的结点，最多也不会超过树的深度。

## 项目目标

本项目即是通过用户输入的一组数据建立一棵二叉排序树，并实现二叉排序树基本的插入和查询功能。

# 实现思路

## 2.1 数据的存储结构

文本

描述已自动生成二叉排序树是基于二叉树衍生出的一种数据结构，所以结点的设计仍旧与【表达式转换】项目中的表达式二叉树保持一致：对于要建立的表达式二叉树，我们采用二叉链表存储，树结点结构如下：

这里为了方便在后续进行二叉树遍历输出，重载了 << 运算符。

此外，需要一个char型变量存储用户输入的操作，一个int型变量存储正在处理的数据以及一个bool类型的变量标记正在处理的数据在二叉排序树中是否重复。

## 2.2 类的设计及关系

这里值得说明的是，BiSortTree类中的find方法和insert方法均是由关键字private修饰的，因为这两个方法是直接对于BiTree类中的root结点进行操作的，且由于是递归实现，为确保程序性能，不方便在其中添加操作安全性的判断。而在public方法中有相应的findElem方法和insertElem方法，在确保输入安全的情况下内置调用了find方法以及insert方法，也是尽可能的采用了面向对象的编程思想。

## 2.3 基本思路

由于二叉排序树是一个动态更新的数据结构，所以在进行插入操作的时候如何维护其有序性即为本项目的关键。根据二叉排序树的递归定义过程，我们通过递归方式对二叉树进行

## 2.4 功能实现

### 2.4.1 功能菜单

文本

描述已自动生成与之前的许多项目一样，BiSortTree也是一个单例，所以将菜单的打印加入到构造函数当中，此外，该部分还包括了所涉及到数据的初始化。

### 2.4.1 插入元素

文本

描述已自动生成private : insert方法

插入元素时，从当前二叉排序树的根结点开始递归，若当前结点为空，则进行将值插入到当前位置；若要插入的值大于当前结点的值，向右子树递归；反之向左子树递归。若在插入过程中遇到结点值与插入值相等的情况，则输出提示语句并退出本次插入操作。

public : insertElem屏幕上有字

描述已自动生成方法

根据repeatFlag的状态来判定是否成功插入数据。

### 2.4.2 查询操作

private : find方法

查询元素的递归方式与插入元素类似文本

描述已自动生成：从当前结点开始递归，若当前结点为空，则查询失败，所要查询的key值不在该二叉排序树中；若要查询的值大于当前结点的值，向右子树递归；反之向左子树递归。若在查询过程中遇到结点值与插入值相等的情况，则将重复标记置为真，再退出本次查询操作。

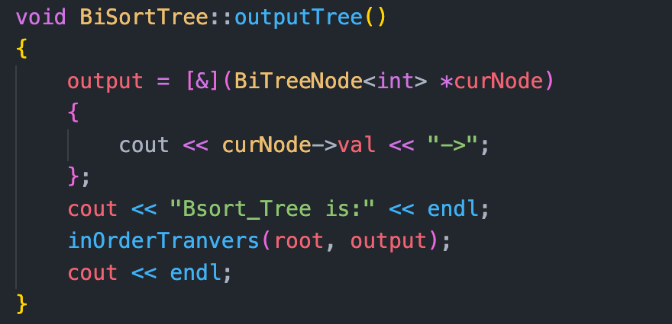
public : findElem方法

根据repeatFlag的状态来判断所查询的数据是否存在与二叉排序树之中屏幕上有字

描述已自动生成。

### 2.4.3 输出二叉排序树

根据二叉排序树的性质，从根结点进行中序遍历即可得到一个有序序列：

outputTree方法首先实现了遍历时对结点进行操作的回调函数output：格式化输出每个结点的值，之后对整棵二叉排序树进行了中序遍历，中序遍历的方式与【表达式转换】中一致：

### 2.4.4 错误处理

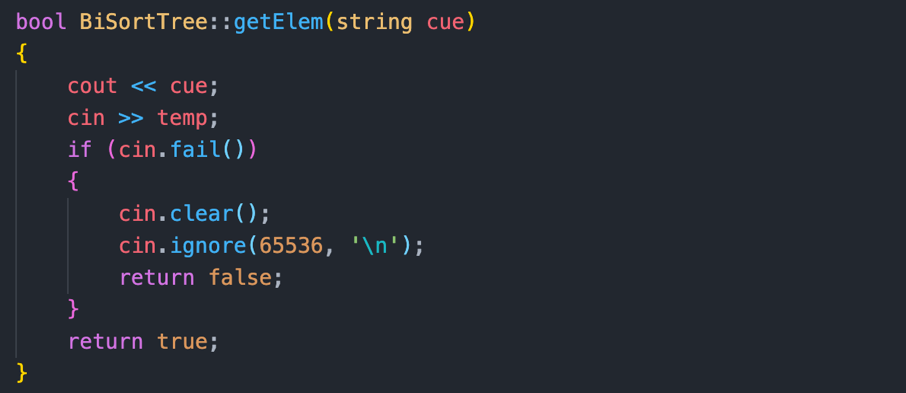
本项目的错误处理策略用户的输入部分：

1. operation code输入：

屏幕上有字

描述已自动生成当且仅当cin状态正确，且输入操作码为有效操作1-4时才判断为输入正确，否则将会输出错误提示并要求重新输入操作码。

1. 二叉排序树的数据输入

在进行多次输入数据建树时：遇到非法输入（由于是int类型数据，所以在超出int范围的数据虽然在数学意义上合法，但由于会置int状态错误，这里仍将其判定为非法输入）时将直接结束建树，建树内容为非法输入之前的所有值。即**非法输入可以作为数据输入结束的标识符。**

在进行单个数据的插入和查询输入时，遇到非法输入会提示输入有误并要求重新输入。

1. 图形用户界面, 文本

   描述已自动生成错误处理实例：

# 性能分析

## 3.1 时间复杂度分析

根据插入和查询的递归过程可以轻易分析得出，本项目二叉排序树的插入和查询操作的时间复杂度为，为二叉排序树的高度。在最坏情况下，二叉排序树可能会建立成为一棵斜树，在插入和查询的时候我们不得不从根结点递归到叶子结点，此时的时间复杂度为，为二叉排序树的结点个数。

## 3.2 优化方向

通过上述分析，二叉排序树的查找性能取决于二叉排序树的形状。因此，如果我们希望对一个集合按二叉排序树查找，最好是把它构建成一棵平衡的二叉排序树。平衡二叉树构建的基本思想就是在构建二叉排序树的过程中，每当插入一个结点时，先检查是否因插入而破坏了树的平衡性，若是，则找出最小不平衡子树。在保持二叉排序树特性的前提下，调整最小不平衡子树中各结点之间的链接关系，进行相应的旋转，使之成为新的平衡子树。

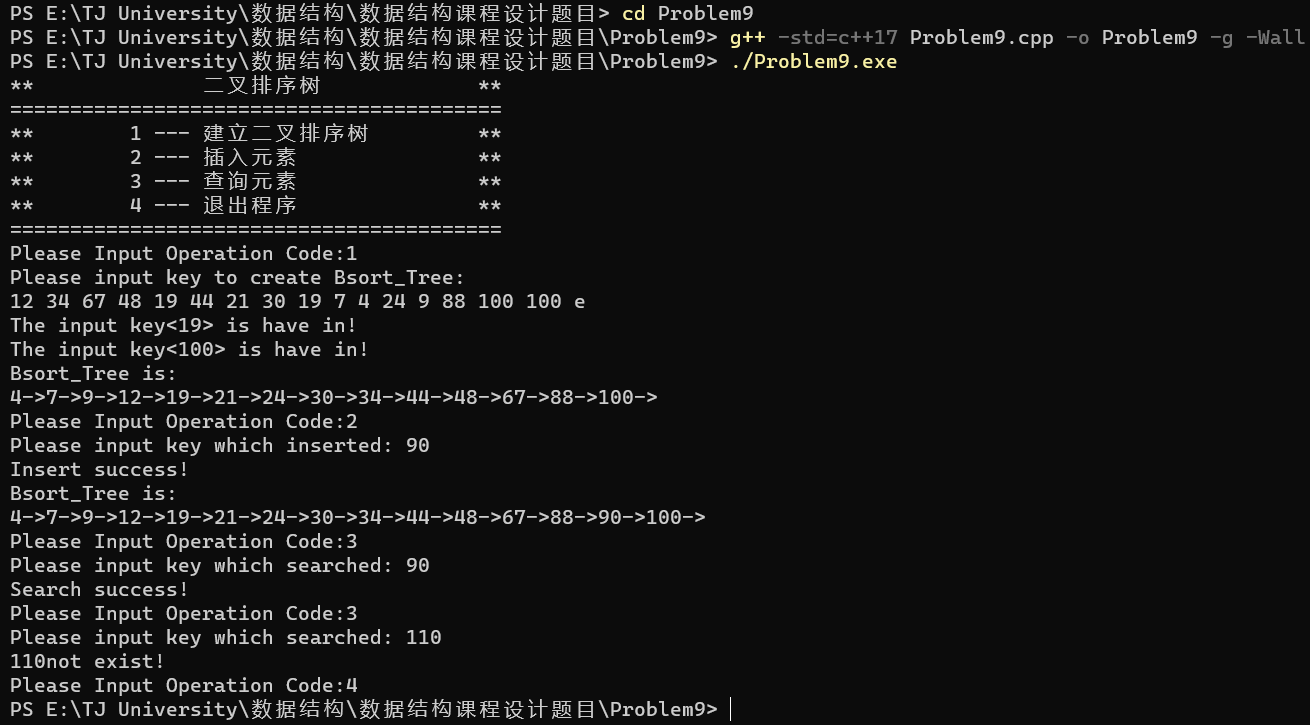
通过这种形式的优化，可以使查询和插入操作的时间复杂度降低至，是一种比较理想的动态查找表算法。

# 四、测试结果

## 4.1 Windows平台

操作系统：Windows11 专业版 21H2

编译器：g++ (GCC) 8.1.0

C++标准：c++17

## 4.2 Linux平台

操作系统：CentOS Linux release 8.5.2111

编译器：g++ (GCC) 8.5.0-4

C++标准：c++17

## 4.3 Mac平台

操作系统：macOS Monterey 12.0.1

编译器：clang++ 13.0.0

C++标准：c++17

