湖南大學

数据结构

课程实验报告

题　　目：特殊二叉树的应用

学生姓名　李晓彤

学生学号　201726010128

专业班级　软件 1701 班

完成日期　2018.11.27

一、 需求分析

0 问题描述

静态查找表（Static Search Table）：只作查找操作的查找表。

动态查找表（Dynamic Search Table）：在查找过程同时插入查找表中不存在的数据元素，或者从查找表中删除已经存在的某个数据元素。

要求：

1.使用二叉查找树（BST）来实现。

2.二叉树使用链式结构（二叉链表）实现。

3.基于 BST，实现静态查找表。

4.在查找表中查找时，要输出关键字比较的次数。

1.问题分析

问题中的需求：

1.二叉树使用链式结构（二叉链表）实现。

2.基于 BST，实现静态查找表。

3.在查找表中查找时，要输出关键字比较的次数。

功能：

1.构造指定结点个数的 BST

2.中序遍历 BST

3.树的深度

4.树的结点个数

5.查找操作

2.输入数据

由问题分析可知：需要处理输入的一系列数据，并且把他们存储成 BST。由 BST 的属性可知：对于二叉检索树的任意一个结点 K，该结点左子树中任意一个结点的值都小于 K；该结点左子树中任意一个结点的值都大于或等于 K。如果按照中序遍历将各个结点打印出来，就会得到由小到大排列的结点。

那么可知问题需要处理输入的结点值和已经存储的结点值的大小关系。

数据输入格式为：

先输入结点个数 n，然后再输入 n 个结点值（每个结点的值用空格隔开）。接下来输入要查找的数据，ctrl+Z 结束查找操作。

例如输入：

5

31 17 42 16 24

31

则表示你输入的一个结点数为 5 的 BST，要查找的值为 31.

3.输出数据

1.先输出已经构造好的 BST：用中序遍历将各个结点打印出来，以此来判断是否构造正确的 BST。

2.输出结点个数和树的深度。

3.根据要查找的元素值输出比较次数。如果没查到则输出：抱歉，您要查找的（元素值）不在该树中,查找一共经过的比较次数为（比较次数）。

如果查到了，则输出：查找成功,查找一共经过的比较次数为（比较次数）

例如输出：

请先输入结点个数(大于 0)：

5

请输入要构造的树:输入 5 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开

例如：31 17 42 16 24 28

31 17 42 16 24

中序遍历：16 17 24 31 42

树的深度：(层数从 0 开始)2

树的结点个数：5

接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序

31

查找成功,查找一共经过的比较次数为： 1

请输入要查找的元素，ctrl+Z 结束程序

43

抱歉，您要查找的 43 不在该树中,查找一共经过的比较次数为： 2

4.测试样例设计

测试样例一：（输入的结点个数为 0）

样例输入：

0

5

31 17 42 16 24

31

17

42

43

样例输出：

------接下来是二叉查找树（BST）实现一个静态查找表------

请先输入结点个数(大于 0)：

0

输入错误，请重新输入：

5

请输入要构造的树:输入 5 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开

例如：31 17 42 16 24 28

31 17 42 16 24

中序遍历：16 17 24 31 42

树的深度：(层数从 0 开始)2

树的结点个数：5

接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序

31

查找成功,查找一共经过的比较次数为： 1

请输入要查找的元素，ctrl+Z 结束程序

17

查找成功,查找一共经过的比较次数为： 2

请输入要查找的元素，ctrl+Z 结束程序

42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
43
抱歉，您要查找的 43 不在该树中,查找一共经过的比较次数为： 2

说明：该测试数据说明了当结点个数为 0 的时候，程序会让用户重新输入结点个数，然后进行接下来的操作。

测试样例二：（普通测试数据）
样例输入：
3
31 17 18
31
27
19
样例输出：
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于 0)：
3
请输入要构造的树:输入 3 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
31 17 18
中序遍历：17 18 31
树的深度：(层数从 0 开始)2
树的结点个数：3
接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序
31
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z 结束程序
27
抱歉，您要查找的 27 不在该树中,查找一共经过的比较次数为： 3
请输入要查找的元素，ctrl+Z 结束程序
19
抱歉，您要查找的 19 不在该树中,查找一共经过的比较次数为： 3

说明：该测试数据测试了当查找数据不在该树中，则程序可以输出查找数据不在该树中这样的提示信息。

测试样例三：（有相同元素，并且相同元素的值大于根结点的值）
样例输入：
9
37 24 42 7 32 40 42 2 120

1
37
42
120

样例输出：
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于 0)：
9
请输入要构造的树:输入 9 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
37 24 42 7 32 40 42 2 120
中序遍历：2 7 24 32 37 40 42 42 120
树的深度：(层数从 0 开始)3
树的结点个数：9
接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序
1
抱歉，您要查找的 1 不在该树中,查找一共经过的比较次数为： 4
请输入要查找的元素，ctrl+Z 结束程序
37
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z 结束程序
42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
120
查找成功,查找一共经过的比较次数为： 4

说明：该组测试数据说明当有相同元素时，并且元素值都大于根结点的值，则存储在 BST 的右子树，而且查找的话找到第一个该元素值就停止查找。

测试样例四：（有相同元素，且元素值小于根结点的值）
样例输入：
5
10 5 9 20 5
5
10
20
9

样例输出：
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于 0)：

5
请输入要构造的树:输入 5 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
10 5 9 20 5
中序遍历：5 5 9 10 20
树的深度：(层数从 0 开始)3
树的结点个数：5
接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序
5
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
10
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z 结束程序
20
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
9
查找成功,查找一共经过的比较次数为： 3

说明：该组测试数据说明当有相同元素时，并且元素值都小于根结点的值，则存储在 BST 的左子树，而且查找的话找到第一个该元素值就停止查找。

测试样例五：（有相同元素，且和根结点的值相同）
样例输入：
5
37 24 42 40 37
37
24
42
40
样例输出：
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于 0)：
5
请输入要构造的树:输入 5 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
37 24 42 40 37
中序遍历：24 37 37 40 42
树的深度：(层数从 0 开始)3
树的结点个数：5
接下来进行查找操作:请输入要查找的元素，ctrl+Z 结束程序

37
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z 结束程序
24
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z 结束程序
40
查找成功,查找一共经过的比较次数为： 3

说明：该组测试数据说明当有相同元素时，并且元素值等于根结点的值，则存储在 BST 的左子树，而且查找的话找到第一个该元素值就停止查找。

二、概要设计

1.抽象数据类型

因为题目要求用二叉链表来实现 BST，并且存储的数据满足：通过 for 循环依次存入，按照存储的先后次序，满足线性特征，即

<ai,ai+1>(0≤a＜n)。

又因为要用 BST 的查找功能，所以用链表会更方便一些，并且题目明确要求用链表存储。

2.算法的基本思想

本题先输入要存储的结点个数，然后利用 BST 自己定义的插入函数构造二叉检索树。查找的时候不需要考虑是空树的情况，因为在构建的时候就已经确定了构建一个非空树，如果是空树是不会构建的。

在查找时：如果一直找到叶子结点都没找到那么说明该元素不在 BST 中，返回-1024，意思是此时的查找次数减 1024，使得查找次数小于 0。如果查找值大于或等于根结点值，那么在右子树中进行查找，进行递归操作，并且查找次数+1，如果查找值小于根结点值，那么在左子树中进行查找，进行递归操作，并且查找次数+1.如果找到了，那么返回 1，然后再递归回去。这里设计的没查找时返回-1024 是方便后续判断查找结果。如果查找结果大于 0，那么说明查找到了，直接输出查找次数，如果查找结果小于 0，那么说明未查找到，输出为查找次数+1024.这样可以抵消没查找到时减去的 1024.

3.程序的流程

程序由两个模块组成：

（1）链表存储模块：

这一模块的功能是：将输入进来的数组构造一个 BST，这一步完成了使用二叉链表实现 BST 的要求。

（2）链表查找模块：

将构建好的 BST 进行查找操作。这里不需要考虑是空树的情况，因为在构建的时候就已经确定了构建一个非空树，如果是空树是不会构建的。如果一直找到叶子结点都没找到那么说明该元素不在 BST 中，返回-1024，意思是此时的查找次数减 1024，使得查找次数小于 0。如果查找值大于或等于根结点值，那么在右子树中进行查找，进行递归操作，并且查找次数+1，如果查找值小于根结点值，那么在左子树中进行查找，进行递归操作，并且查找次数+1.

如果找到了，那么返回 1，然后再递归回去。这里设计的没查找时返回-1024 是方便后续判断查找结果。如果查找结果大于 0，那么说明查找到了，直接输出查找次数，如果查找结果小于 0，那么说明未查找到，输出为查找次数+1024.这样可以抵消没查找到时减去的 1024.

三、详细设计

1.物理数据类型

数据是一串数字，根据这串数字来构建 BST，因为输入的数据结构特性满足线性特性，并且之后要利用 BST 的特性来查找元素，所以选择链表来存储数据，这里的链表继承了结点类。

抽象数据类型的具体设计：

```
ADT   IntegerSet {
      数据对象：D = { ai | ai Î 整数，i = 1,2，…, n, n>=0}
      数据关系：R = {< ai-1，ai> | ai-1, ai ÎD}
      成员：
      int it;//结点元素值
      BiNode* lc;//左孩子指针
      BiNode* rc; //右孩子指针
      BiNode* root; // Root of the BinTree BinTree 树的根结点
      int nodeCount;//结点个数
      基本操作：
       void clear(BiNode* node);//清除操作
       BiNode* getRoot();//BiNode 指针类的返回根结点

       //插入+创建操作
       void insert(const int );
       BiNode* inserthelp(BiNode* root,const int);

       int find(BiNode*,const int);//查找操作

       //中序遍历
       void Inorder(BiNode* rt);

       //二叉树深度
       int BiTreeDepth(BiNode* rt);

       //节点数统计
       int count(BiNode* rt);
```

2.输入和输出的格式

输入流程及格式：输入的第一个数字为结点个数，接下来的 n 个值为要构建的二叉检索树的结点值，中间用空格隔开，回车结束输入

输出流程：

先输出系统名称：------接下来是二叉查找树（BST）实现一个静态查找表------

然后输出提示信息和输入例子：

请先输入结点个数(大于 0)：

当用户输入结点个数后，输出提示信息和输入例子：

请输入要构造的树:输入 n 个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开，例如：31 17 42 16 24 28

当用户输入完结点值后，输出中序遍历结果，以此来证明是否成功构建 BST。然后输出树的基本属性：树的结点值和树的深度。

接下来用户输入要查找的值，然后进行查找操作，如果找到了输出查找成功的信息，以及比较次数。如果未查找到，则输出查找失败的信息和比较次数。

3.算法的具体步骤

1.先输入结点个数来构建二叉检索树。

2.输入对应的结点值，并且利用自身定义的成员函数来完成二叉检索树的构造。

3.进行中序遍历，来检查是否构建正确的二叉检索树。

4.进行查找操作，利用 while 循环来让用户输出查找值，如果没找到则输出查找失败信息，如果找到了则输出查找成功的信息。这里是利用二叉检索树本身的成员函数来完成查找操作。

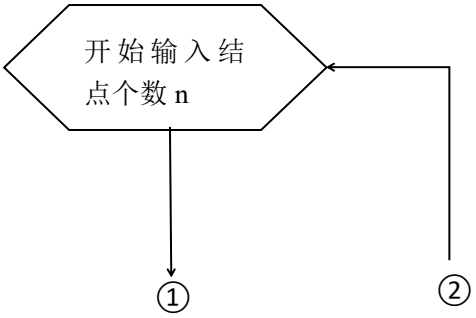
5.输出查找需要的比较次数。

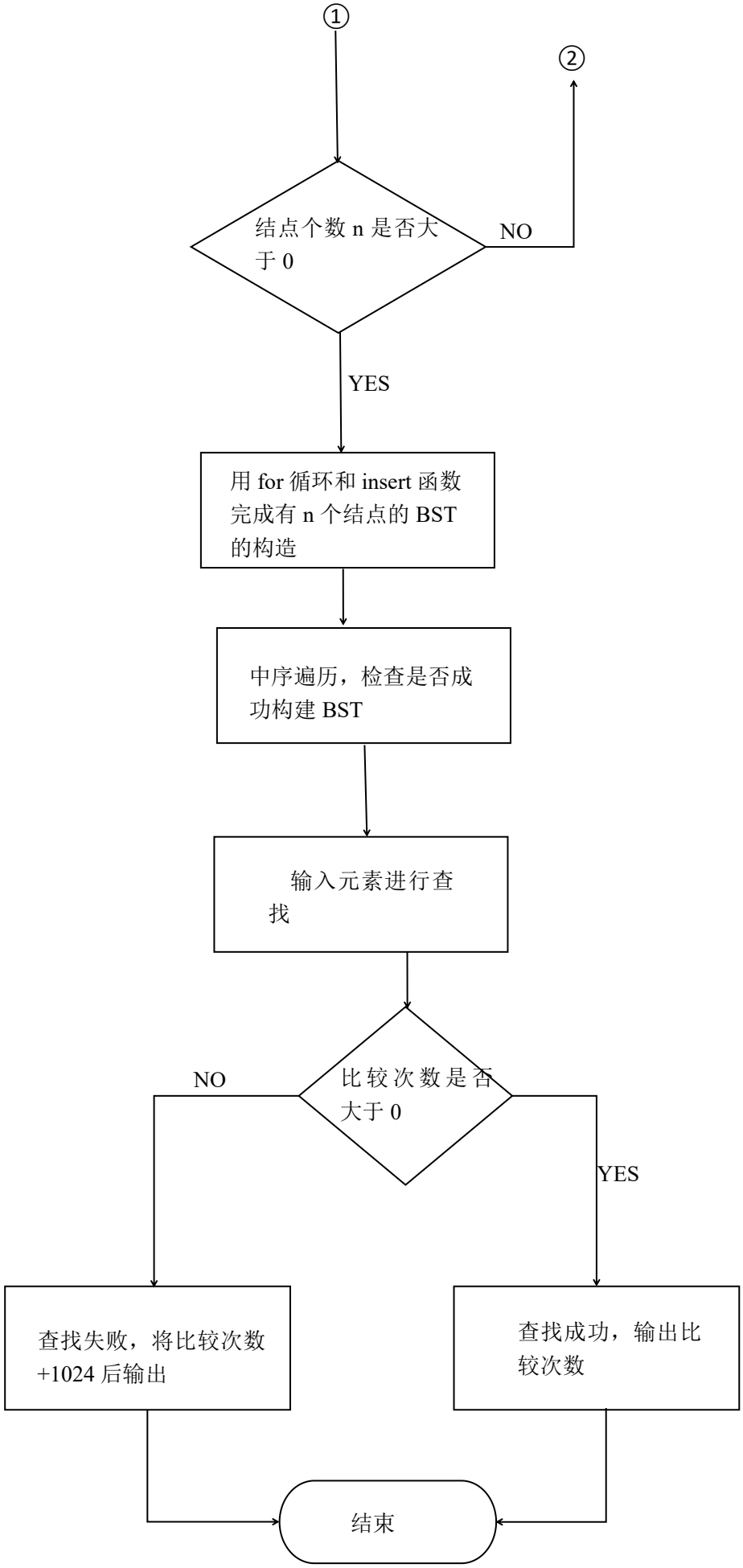
伪代码：

```
int main(int argc, char** argv) {
    int depth=0;//树的深度
    int num=0;//树的结点个数
    BST T;//二叉检索树类的实例化
    cout<<"------接下来是二叉查找树（BST）实现一个静态查找表------"<<endl;
    cout<<"请先输入结点个数(大于 0)："<<endl;
    cin>>num;
    while(num<=0)//当输入的结点个数小于 0 则说明输入错误，重新输入
    {
        cout<<"输入错误，请重新输入："<<endl;
        cin>>num;
    }

//接下来开始构建二叉检索树
    cout<<"请输入要构造的树:";
    cout<<"输入"<<num<<"个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开"<<endl;
    cout<<"例如：31 17 42 16 24 28"<<endl;
    int temp=0;
    //输入+建树
    for(int i=0;i<num;i++)
    {
        cin>>temp;
        T.insert(temp);//利用自身定义的成员函数来完成二叉检索树的构造
    }

//中序遍历检查是否构建正确的二叉检索树
    cout<<"中序遍历：";
```

```
    T.Inorder(T.getRoot());
    cout<<endl;

//二叉检索树树的基本信息
    cout<<"树的深度：(层数从 0 开始)";
    depth=T.BiTreeDepth(T.getRoot());
    cout<<--depth<<endl;
    cout<<"树的结点个数：";
    num=T.count(T.getRoot());
    cout<<num<<endl;

//接下来是查找模块
    cout<<"接下来进行查找操作:";
    cout<<"请输入要查找的元素，ctrl+Z 结束程序"<<endl;
    while(cin>>temp)//输入要查找的值，ctrl+Z 结束程序
    {
        //利用二叉检索树本身的成员函数来完成查找操作
        int bijiao=T.find(T.getRoot(),temp);

        if(bijiao<=0)//比较次数小于 0，说明没有查找到
        {
            cout<<"抱歉,您要查找的"<<temp<<"不在该树中"<<",查找一共经过的比
较次数为：  "<<bijiao+1024<<endl;
        }
        if(bijiao>0)//如果比较次数大于 0 说明找到了
        {
            cout<<"查找成功"<<",查找一共经过的比较次数为：  "<<bijiao<<endl;
        }
        cout<<"请输入要查找的元素，ctrl+Z 结束程序"<<endl;//进行下一轮查找
    }

}
```

流程图：




①
②
结点个数 n 是否大于 0
NO
YES
用 for 循环和 insert 函数完成有 n 个结点的 BST 的构造
中序遍历，检查是否成功构建 BST
输入元素进行查找
比较次数是否大于 0
NO
YES
查找失败，将比较次数+1024 后输出
查找成功，输出比较次数
结束

4.算法的时空分析

（1）链表存储模块：

由于是用 for 循环和 insert 函数存储，具体由输入数据规模决定。Insert 的时间代价取决于结点插入的深度。这样这个操作中的最差情况都等于该树的深度。如果二叉树平衡，则有 n 个结点的二叉树的高度约为 log n，但是如果二叉树完全不平衡（如形成一个链表的形状），则高度可以达到 n。因而，平衡二叉树每次插入的平均时间代价为Θ(log n)，而严重不平衡的 n 个结点的 BST 在最差情况下的时间代价为Θ(n)。如果按照逐个插入的方式构建一棵有 n 个结点的 BST，并且很幸运的遇到了每个结点的插入都使输保持平衡的情况，那么每次插入时间为Θ(log n)，共为Θ(n log n)。但是如果结点按照递增顺序插入，就会得到一条高度为 n 的链，插入时间代价为Θ(n^2)。

（2）链表查找模块：

由于是用 while 循环以及 find 函数完成查找操作，具体查找次数由输入的待查找数据个数决定，而 find 函数的时间代价取决于结点结点被找到插入的深度。这样这个操作中的最差情况都等于该树的深度。如果二叉树平衡，则有 n 个结点的二叉树的高度约为 log n，但是如果二叉树完全不平衡（如形成一个链表的形状），则高度可以达到 n。因而，平衡二叉树每次查找的平均时间代价为Θ(log n)，而严重不平衡的 n 个结点的 BST 在最差情况下的时间代价为Θ(n)。如果按照逐个插入的方式构建一棵有 n 个结点的 BST，并且很幸运的遇到了每个结点的插入都使输保持平衡的情况，那么每次查找时间为Θ(log n)，共为Θ(n log n)。但是如果结点按照递增顺序插入，就会得到一条高度为 n 的链，那么查找时间代价为Θ(n^2)。

四、调试分析

1.调试方案设计

调试目的：为了发现代码是否有语法错误、连接错误、逻辑错误和运算错误，有不严谨之处。

样例：

```
0
5
31 17 42 16 24
31
17
42
43
```

设置断点处：链表存储模块中调用 insert 函数处

调试计划：留心观察是否可以成功插入结点，并且按照 BST 的性质插入。

设置观察变量：root，val；

2. 调试过程和结果，及分析

调试过程：如果此时结点为 NULL，则新建结点，把 val 的值存入。如果此时存入的值大于或等于结点值，那么遍历结点的右子树，然后找到合适的位置存储；如果此时存入的值小于结点值，那么遍历结点的左子树，然后找到合适的位置存储。

问题：在进入 inserthelp（）函数中，直接跳过了判断结点是否是 NULL，来执行接下来的遍历右子树或者左子树的操作，造成指针越界问题。

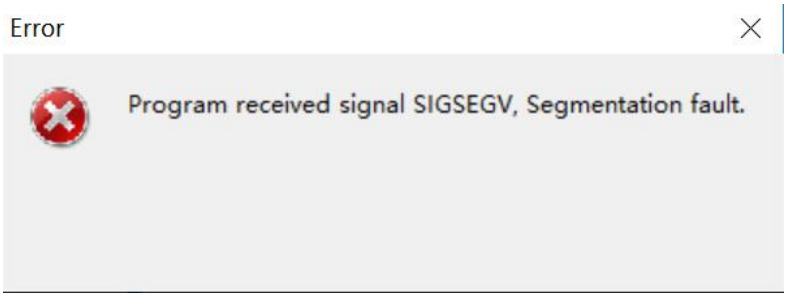
程序出现“Program received signal SIGSEGV，Segmentation fault.”的报错信息，程序崩溃电脑黑屏。

解决方法：重写 BST 和 BiNode 的构造函数和析构函数。

五、测试结果

测试样例一：（输入结点个数为 0 后程序的处理情况）

错误输出：



正确输出：

```
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于0)：
0
输入错误，请重新输入：
5
请输入要构造的树:输入5个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
31 17 42 16 24
中序遍历：16 17 24 31 42
树的深度：(层数从0开始)2
树的结点个数：5
接下来进行查找操作:请输入要查找的元素，ctrl+Z结束程序
31
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z结束程序
17
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
43
抱歉，您要查找的43不在该树中,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
^Z
请按任意键继续. . .
```

分析结论：当输入的结点个数为 0 时会让用户重新输入结点个数，这就保证了不会出现空树的情况。

测试样例二：（一般情况）

```
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于0)：
3
请输入要构造的树:输入3个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
31 17 18
中序遍历：17 18 31
树的深度：(层数从0开始)2
树的结点个数：3
接下来进行查找操作:请输入要查找的元素，ctrl+Z结束程序
31
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z结束程序
27
抱歉，您要查找的27不在该树中,查找一共经过的比较次数为： 3
请输入要查找的元素，ctrl+Z结束程序
19
抱歉，您要查找的19不在该树中,查找一共经过的比较次数为： 3
请输入要查找的元素，ctrl+Z结束程序
^Z
请按任意键继续. . .
```

分析结论：当输入一组数据后，按照 BST 存储规则构造 BST 树，然后进行查找操作。

测试样例三：（特殊情况：有相同元素，且比根结点的值大）

```
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于0)：
9
请输入要构造的树:输入9个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
37 24 42 7 32 40 42 2 120
中序遍历：2 7 24 32 37 40 42 42 120
树的深度：(层数从0开始)3
树的结点个数：9
接下来进行查找操作:请输入要查找的元素，ctrl+Z结束程序
1
抱歉，您要查找的1不在该树中,查找一共经过的比较次数为： 4
请输入要查找的元素，ctrl+Z结束程序
37
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z结束程序
42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
120
查找成功,查找一共经过的比较次数为： 4
请输入要查找的元素，ctrl+Z结束程序
^Z
请按任意键继续. . .
```

分析结论：当有相同元素，且比根结点的值大时，则存储到右子树中，然后再执行查找操作。这时查找相同元素时，找到第一个就停止查找操作。

测试样例四：（特殊情况：有相同元素，且比根结点的值小）

```
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于0)：
5
请输入要构造的树:输入5个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
10 5 9 20 5
中序遍历：5 5 9 10 20
树的深度：(层数从0开始)3
树的结点个数：5
接下来进行查找操作:请输入要查找的元素，ctrl+Z结束程序
5
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
10
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z结束程序
20
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
9
查找成功,查找一共经过的比较次数为： 3
请输入要查找的元素，ctrl+Z结束程序
^Z
请按任意键继续. . .
```

分析结论：当有相同元素，且比根结点的值小时，则存储到左子树中，然后再执行查找操作。这时查找相同元素时，找到第一个就停止查找操作。

测试样例五：（特殊情况：有相同元素，且和根结点的值相同）

```
------接下来是二叉查找树（BST）实现一个静态查找表------
请先输入结点个数(大于0)：
5
请输入要构造的树:输入5个结点，请先输入要构造的根结点，之后的顺序随意，每个结点的值用空格隔开
例如：31 17 42 16 24 28
37 24 42 40 37
中序遍历：24 37 37 40 42
树的深度：(层数从0开始)3
树的结点个数：5
接下来进行查找操作:请输入要查找的元素，ctrl+Z结束程序
37
查找成功,查找一共经过的比较次数为： 1
请输入要查找的元素，ctrl+Z结束程序
24
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
42
查找成功,查找一共经过的比较次数为： 2
请输入要查找的元素，ctrl+Z结束程序
40
查找成功,查找一共经过的比较次数为： 3
请输入要查找的元素，ctrl+Z结束程序
^Z
请按任意键继续. . .
```

分析结论：当有相同元素，且等于根结点时，则存储到左子树中，然后再执行查找操作。这时查找相同元素时，找到第一个就停止查找操作。

六、实验日志

2018/11/14

内容：实验四上机

确定了改造二叉链表实现 BST 的思想。设计了链表 ADT。

2018/11/25

内容：确定查找 find 函数

1.find 函数：

初想法：定义一个整型变量 bijiao，来存储比较的次数。

遇到的问题：当是空子树的时候不知道怎么返回比较次数。或者没找到结点值，则不知道如何区分空子树的情况。

修改后的想法：改变输入格式。让原先以/结尾的输入格式变成了先输入结点值然后再输入数据值。这样就可以解决空子树问题，然后如果查找不到，则让查找次数-1024，这样减的值很大，可以确保最后返回的查找次数为负，然后就可以判断是否找到。

2.输入：

初想法：定义一个字符数组来存储输入的值以及结束输入的/

遇到的问题：当把 char 类型转化为 int 类型，在调试过程中发现，如果输入的是两位及以上的整数，那么 char 只会存第一位的数字。

修改后的想法：让原先以/结尾的输入格式变成了先输入结点值然后再输入数据值。改用 for 循环和 insert 直接构造 BST 树，摒弃了先存值后建树的思想。

2018/11/26

内容：确定插入 insert 函数

查找：

1.初想法：先判断结点是否为 NULL，若为 NULL 则建立新的结点。

遇到的问题：进入 inserthelp（）函数后，直接跳过了判断结点是否为 NULL，直接进行下

面的比较操作，造成指针越界问题。
修改后的想法：改变 BST 的构造函数，使得 BST 实例化的结点为 NULL。然后建立新结点的时候直接用 new。

经验和体会：在使用链表时要注意指针越界问题；这次遇到的主要问题就是传参。刚开始不知道怎么传参来构造 BST，后面又不知道怎么写 find 函数的接收参数形式。很幸运有同学帮助，讲解后明了了许多。通过本次实验，更加了解了 BST 的使用方法和应用。