
南開大學
Nankai University

# 南　开　大　学

## 网 络 空 间 安 全 学 院

## 数 据 结 构 实 验 报 告

---

**第五次实验作业**

---

罗功成 1910487

年级：2019 级

专业：信息安全

指导教师：王玮

2021 年 12 月 15 日

# 摘要

根据给定数组，构建二叉搜索树，并打印相应树形结构和输出指定节点。


**关键字：链表，二叉搜索树，树形结构，树的遍历**


# 目录

# 一、 实验流程

## (一) 实验要求

1. 根据数值不重复的数组值，以链表形式逐点插入实现二叉搜索树，并打印对应的结构图。
2. 输出指定节点的后序遍历的后继节点
3. 替换所有节点值为树中所有小于该点的节点值之和，

## (二) 实验原理和思路

1. 打印树形结构：在树不为空时，向下查看树的左孩子和右孩子；当有左孩子时，就输出 endl 换行，并且在下一行跟上一个“|”，表示进入到孩子节点处，在换一行，在‘|’下打印对应的元素；再看右孩子，如果有，就接在左孩子后面，中间用”————” 分开即可。最后再递归调用，将孩子节点作为新的当前节点继续向下查看，直到最后遍历完成。
2. 输出后序遍历后继节点：后序遍历是左孩子-右孩子-父节点的方式，所以考虑递归调用，进入后序遍历后，先递归调用以左孩子为当前节点的，再递归以右孩子为当前节点的，最后再打印当前的对应节点。这里要注意用一个数组 X 来保存对应的各节点数据。到主函数调用时，直接在按顺序保存了后序遍历的数组里进行查找。输入要查找的元素 temp，当 temp 和 a[i] 匹配时，a[i+1] 就是后序遍历的后继节点。
3. 将元素替换为小于该元素的元素之和：为了解决数据的覆盖问题，可以再开一个和原数组 a 一样大的数组 b，各位置初始化为 0. 之后进入两重循环当中，一重是对 a[i] 的遍历，将当前 a[i] 与数组中其他的元素值进行比较，小于该元素的将其累加到 b[j] 中，最后 j 累加 1 直至 b 数组结束。然后以 b 数组中的数值作为新的输入构建二叉树，然后对其打印即可。

## (三) 实验核心代码

二叉搜索树的链表实现

```
#include <iostream>
using namespace std;
template<typename T>
//树结点结构
class BSTNode {
public:
    T _key;
    BSTNode* _lchild;
    BSTNode* _rchild;
    BSTNode* _parent;

    //构造函数
    BSTNode(T key, BSTNode* lchild, BSTNode* rchild, BSTNode* parent) :
        _key(key), _lchild(lchild), _rchild(rchild), _parent(parent) {};
};


template<typename T>
```

```
class BSTree {
private:
    BSTNode<T>* _Root;   //根结点

public:
    BSTree() :_Root(NULL) {};
    ~BSTree() {};

    void insert(T key);//二叉树的插入
    BSTNode<T>* search(T key);//二叉树的查找

    void preOrder();   //先序输出
    void inOrder();    //中序输出
    void postOrder(); //后序输出

    void print();//打印二叉树
    void remove(T key);
    BSTNode<T>* sucessor(BSTNode<T>* x); //查找某个结点的后继

};
```

打印对应的树形结构

```
template<typename T>
void BSTree<T>::print(BSTNode<T>*& tree)
{

    if (tree) //如果tree不为空
    {

        if (tree->_lchild) //结点有左孩子
        {
            cout << endl;
            cout << "|" << endl;
            cout << tree->_lchild->_key;
        }


        if (tree->_rchild)
        {

            cout<<"---";//注意，这里用\\来表示一个\，否则 单个的\是转义字符。
            cout << tree->_rchild->_key;

        }

        print(tree->_lchild);
        print(tree->_rchild);

```

```
    }
}
```

后继遍历实现

```
int x[8]; int i = 0;//注意，头文件里直接生成的全局变量在cpp中也可以直接使用。
template <typename T>
void BSTree<T>::postOrder(BSTNode<T>*& tree) const
{
    if (tree)
    {
        postOrder(tree->_lchild);
        postOrder(tree->_rchild);
        cout << tree->_key << " ";

        x[i] = tree->_key;
        i++;
    }
}

int temp;
        cout << "输入想要查看的节点" << endl;
        cin>>temp;//要找到的元素
        for(int i=0;i<7;i++)
        {
        if(x[i]==temp)
            cout<<"后序遍历的后继节点为"<<x[i+1]<<endl;
        }
```

元素替换

```
    int b[8];
    for (int i = 0; i < 8; i++)
    {
        b[i] = 0;
    }

    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            if (a[i] >= a[j])
                b[i] = b[i] + a[j];
        }
    }

    for (int i = 0; i < 8; i++)
    {
        cout << b[i] << endl;
```

```
19          s1.insert(b[i]);
20
21      }
```

## （四） 实验结果与分析

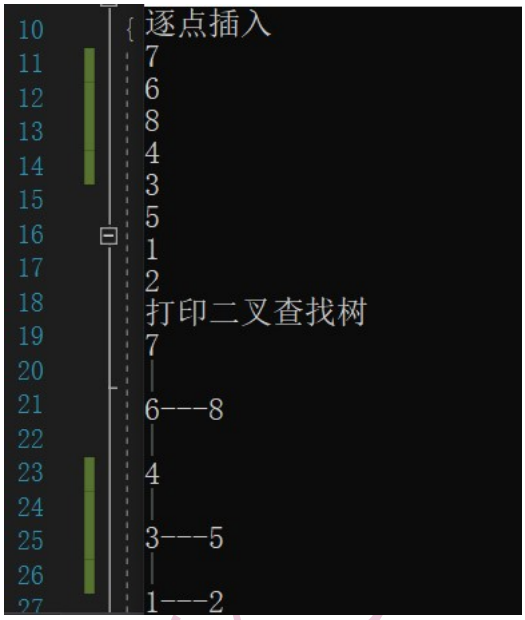


图 1: 打印二叉树树形结构

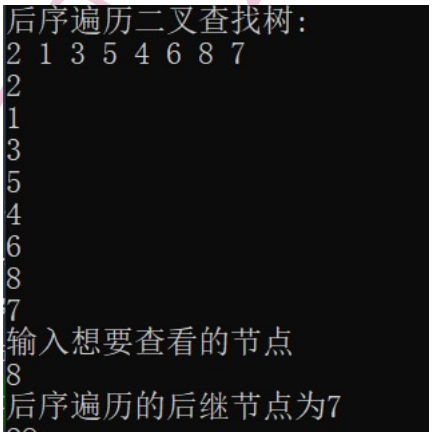


图 2: 输出指定节点的后序遍历后的后继节点

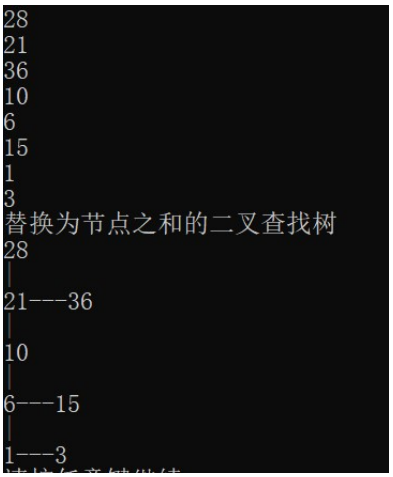


图 3: 替换为元素之和的二叉树

可见，以上代码可以满足本次实验的各项要求。

## 二、 实验探究问题解析

时间复杂度：O（n2）

插入二叉树各个元素过程为 O（n），对二叉树的后序遍历查找取决于是否是构造的是平衡二叉树，所以介于 O（logn）到 O（n）之间，在元素替换阶段每一个 b[i] 对 a[i] 都需要进行一次遍历，两者总数均为 n, 所以复杂度为 O(n2). 综上为 O（n2）

空间复杂度：O(n)

在比较元素大小来决定是否累加，输入想要查找的元素用到的两处 temp 只是中间变量，不影响空间复杂度；

开始阶段用数组 a 保存输入的各个元素，在中间保存后序遍历的各个节点的过程中用到数组 x，在最后完成元素替换过程用到数组 b，大小均为 n, 所以总体的空间复杂度也是线性表的复杂度，即 O(n)。

SP=n*(addr(a)+addr(x)+addr(b)+sizeof(int)).

## 三、 总结和收获

1. 对二叉搜索树的链表实现方法，二叉树的遍历，查找对应节点，节点替换等相关知识得到了巩固和提高。

2. 对链表，指针，类的编写和实例化，函数调用等编程能力得到提高。