河南工業大學

《算法设计与分析》综合性实验
实验报告

题　　目：二叉树插入

姓　　名：田劲锋

班　　级：软件 1305 班

学　　号：201316920311

指导教师：靳小波

完成时间：2014 年 11 月 28 日

一、 实验题目

二叉树的插入

二、 实验目的

1. 实现二叉查找树的插入

三、 实验要求

给定一系列数，依次插入到二叉树中。

```
5
12 5 24 13 46
```

四、 程序流程图

二叉查找树[1]，是一棵空树，或是具有下列性质的二叉树：

1. 若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值；
2. 若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值；
3. 它的左、右子树也分别为二叉查找树。

二叉查找树是递归定义的，如图1是一棵二叉查找树。

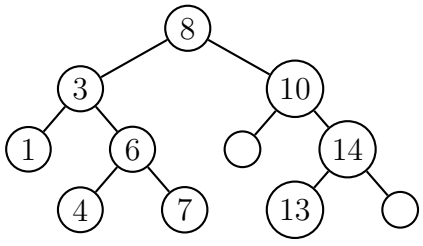

图 1: 一棵三层的二叉查找树

我们用链表结构来存储二叉查找树，其中每一个节点都是一个对象。有$data$存储其值，$left$和$right$指向其左右儿子。

[1]本节内容来源于我之前的论文[1]。

对于一个已知的二叉查找树，从小到大输出其节点的值，只需对其进行二叉树的中序遍历，即递归地先输出其左子树，再输出其本身，然后输出其右子树。遍历的时间复杂度为$O(n)$。这里，我们给出这一算法的伪代码，参见图2。

```
INORDER(x)
1  if x ≠ NIL
2        INORDER(x.left)
3        输出 x.data
4        INORDER(x.right)
```

图 2: 中序遍历输出

将一个新值v插入到树T中，基本方法是类似于线性表中的二分查找，不断地在树中缩小范围定位，最终找到一个合适的位置插入。具体方法如下所述：

1. 从根节点开始插入；
2. 如果要插入的值小于等于当前节点的值，在当前节点的左子树中插入；
3. 如果要插入的值大于当前节点的值，在当前节点的右子树中插入；
4. 如果当前节点为空节点，在此建立新的节点，该节点的值为要插入的值，左右子树为空，插入成功。

对于相同的元素，一种方法我们规定把它插入左边，另一种方法是我们在节点上再加一个域，记录重复节点的个数。上述方法为前者。插入的时间复杂度仍为$O(h)$。算法的伪代码参见图4。

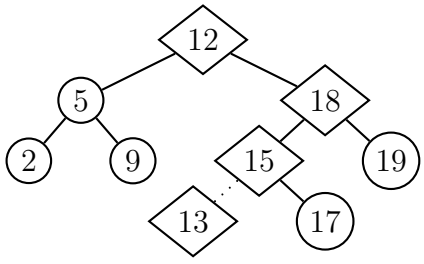

图 3: 将13插入一棵二叉查找树。菱形表示从根节点开始向下到插入位置的路径，虚线为在树中插入该节点而产生的链接。

五、 程序代码

```
1 #include <iostream>
```

INSERT(T, v)

1 **if** $T = \text{NIL}$

2 new $T.key = v$ // 新建节点并插入

3 **elseif** $v \leq T.key$

4 $T.left.p = T$

5 INSERT$(T.left, v)$

6 **else**

7 $T.right.p = T$

8 INSERT$(T.right, v)$

图 4: 二叉查找树的插入

```
using namespace std;

template <class ElemType>
class BSTree
{
private:
    class BSTNode
    {
    public:
        BSTNode* left;
        BSTNode* right;
        ElemType data;
        BSTNode() :left(NULL), right(NULL) {}
        BSTNode(ElemType _data) :data(_data), left(NULL), right(NULL) {}
    };
    typedef BSTNode* NodeP;
    NodeP root;

public:
    BSTree() :root(NULL) {}
    ~BSTree() { delete_node(root); }
    bool empty() const { return root == NULL; }
    void insert(const ElemType &_data) { insert_node(root, _data); }
    void inorder(ostream &out) const { inorder_node(out, root); }

protected:

    inline void delete_node(NodeP _node)
    {
        if (_node->left != NULL) {
            delete_node(_node->left);
        } else if (_node->right != NULL) {
            delete_node(_node->right);
        } else if (_node != NULL) {
```

```
36              delete _node;
37              _node = NULL;
38          }
39      }
40
41      inline void insert_node(NodeP &_root, const ElemType &_data)
42      {
43          if (_root == NULL) {
44              _root = new BSTNode(_data);
45          } else if (_data <= _root->data) {
46              /* 小的放左边 */
47              insert_node(_root->left, _data);
48          } else {
49              /* 大的放右边 */
50              insert_node(_root->right, _data);
51          }
52      }
53
54      inline void inorder_node(ostream &out, NodeP _root) const
55      {
56          if (_root != NULL) {
57              inorder_node(out, _root->left); // 左
58              out << _root->data << " "; // 根
59              inorder_node(out, _root->right);// 右
60          }
61      }
62  };
63
64  int main()
65  {
66      BSTree<int> T;
67      int n, a;
68
69      cin >> n;
70      for (int i = 0; i < n; i++) {
71          cin >> a;
72          T.insert(a);
73      }
74      T.inorder(cout);
75      cout << endl;
76
77      return 0;
78  }
```

六、 实验结果

输出文件output.txt：

```
5 12 13 24 46
```

七、 实验体会

该实验题目说是二叉树的插入，其实准确来讲应该是二叉查找树的插入。关于二叉查找树的问题，无数先人已经探索出来了非常平整的道路了。包括随之拓展出来的二叉平衡树，也是充满了数学之美。三年前我也写过这样的论文，当然并没有什么创新价值，所以就没有发表，不过网络上应该是可以搜索到的。关于二叉树的问题我已经研究过很多了，随之产生的思考也是无穷尽的。

至于程序流程图，我还是一如既往地不画了。描述算法的最好办法依然是伪代码，没有人是用复杂的流程图来描述算法的。

程序是C++编写的，也是为了熟悉面向对象的程序设计方法。我对C++也不是很熟悉，调试也欠缺不少，比如如下的编译警告我暂时还没能解决，希望今后能找到解决办法。

```
alys02.cpp: In constructor `BSTree<ElemType>::BSTNode::BSTNode(ElemType) [with E
lemType = int]':
alys02.cpp:44:  instantiated from `void BSTree<ElemType>::insert_node(BSTree<El
emType>::BSTNode*&, const ElemType&) [with ElemType = int]'
alys02.cpp:24:  instantiated from `void BSTree<ElemType>::insert(const ElemType
&) [with ElemType = int]'
alys02.cpp:72:   instantiated from here
alys02.cpp:13: warning: `BSTree<int>::BSTNode::data' will be initialized after
alys02.cpp:11: warning:  `BSTree<int>::BSTNode*BSTree<int>::BSTNode::left'
alys02.cpp:15: warning:   when initialized here
```

参考文献

[1] 由BST到SBT, 田劲锋, 2011