中 山 大 学 软 件 学 院 软 件 工 程 专 业 **2010** 级 (2010 学年秋季学期)

《SE-203 数据结构与算法》 期 末 试 题 参 考 答 案 (B)

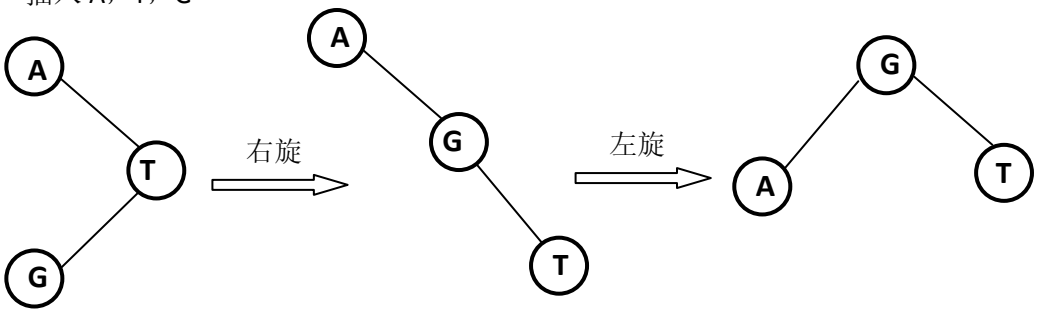
I. 单项选择 (每小题 2 分，共 30 分)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	C	B	A	D	D	A	B	A	D	D	A	B	B	C

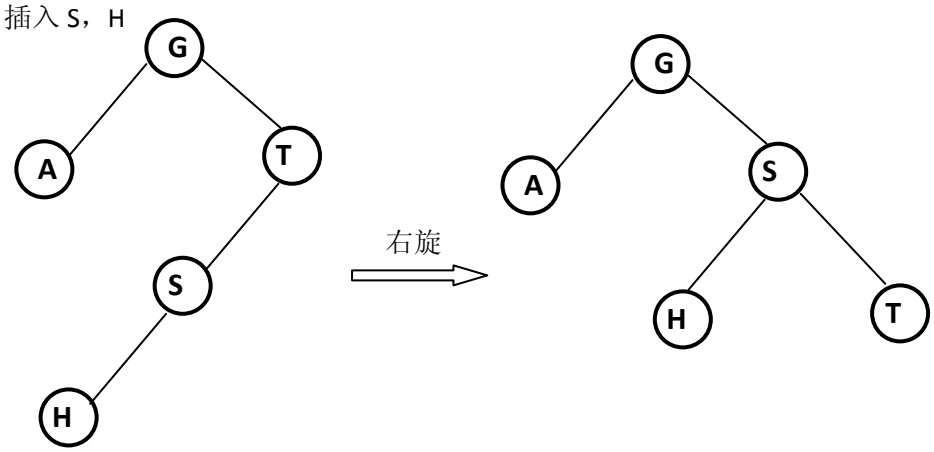
II. 解答题 (每小题 15 分，共 45 分)

1. AVL

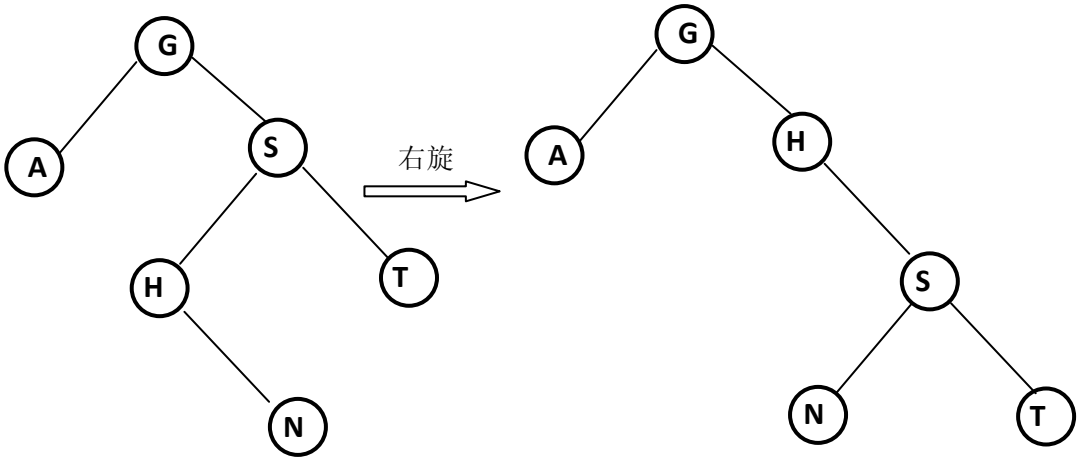
插入 A，T，G

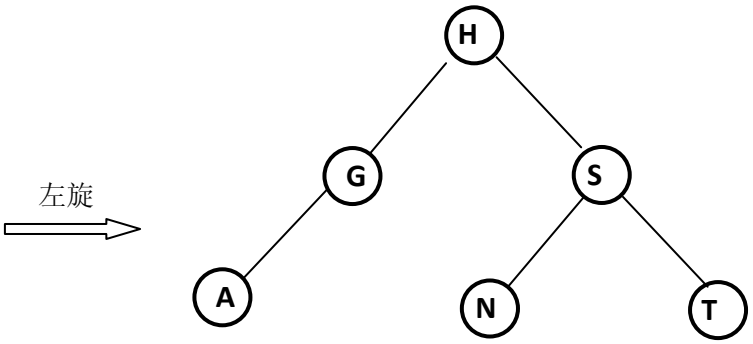


插入 S，H



插入 N

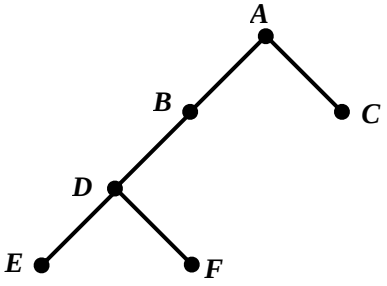




2. 无向图的邻接矩阵如下：

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	0	1	1	0
C	1	0	0	0	0	0
D	1	1	0	0	1	1
E	0	1	0	1	0	0
F	1	0	0	1	0	0

深度优先搜索树如下：



3. 快速排序算法是一种“分而治之”的排序思想。假设待排序的数据存储在 Data[low..high]之中，其排序思想如下：

(1). 如果待排数据的区间是错误的，则排序结束；

(2). 如果 high–low+1<2，则排序结束；

(3). 根据支点的选择策略确定一个支点；

(4). 利用支点把待排数据分成“比支点小的部分”、支点(m 是支点的位置)和“比支点大的部分”；

(5) 递归地对待排数据 Data[low..m–1]和 Data[m+1,high]进行快速排序；

(6) 已排好序的数据 Data[low..m–1]、Data[m+1,high]和支点 Data[m]组合在一起，得到到一个已排好序的数据 Data[low..high]。

对题中给定数据的排序步骤如下：

第一次支点：4

第一次划分: 1, 3, 2, 4, 6, 5, 7

第二次支点: 1

第二次划分: 1, 3, 2

第三次支点: 3

第三次划分: 1, 2, 3

第四次支点: 6

第四次划分: 5, 6, 7

最终结果: 1, 2, 3, 4, 5, 6, 7

如果上述步骤表述基本正确，就算解答正确。

III. 编程题 (共 25 分)

1. 参考程序

(1) 递归版本

```
template <class Record>
Binary_node<Record> *Search_tree<Record>::
                    search_for_node( Binary_node<Record> *sub_root,
                                                    const Record & target) const
{
        if (sub_root == NULL || sub_root->data == target)
                return sub_root;
        else if (sub_root->data < target)
                return search_for_node(sub_root-> right, target);
        else return search_for_node(sub_root->left, target);
}
```

(2) 非递归函数

```
template<class Record>
Binary_node<Record> *Search_tree<Record> search_for_node(Binary_node<Record>
*sub_root, const Record &target)
{
while(sub_root != NULL && sub_root->data != target)
{
   if(sub_root->data < target)
         sub_root = sub_root->right;
   else
         sub_root = sub_root->left;
}

   return sub_root;
}
```

2. 参考程序

(1)

```
void Set::operator -= (const Set &Src)
{
   Node *head, *Pt, *Pt1, *Pt2;
   head = NULL;
   for (Pt = Head; Pt != NULL; Pt = Pt->next) {
        for (Pt2 = Src.Head; Pt2 != NULL; Pt2 = Pt2->next)
             if (Pt->Key == Pt2->Key) break;
        if (Pt2 != NULL) continue;
        Pt2 = new Node(Pt->Key);
        if (Pt2 != NULL) {
           if (head == NULL) head = Pt2;
           else Pt1->next = Pt2;
           Pt1 = Pt2;
        }
   }
   while (Head != NULL) { Pt = Head->next;      delete Head;    Head = Pt; }
   Head = head;
}
```

(2)

```
int Set:: Counter ()
{
   int count = 0;
   for (Node *Pt = Head; Pt != NULL; Pt = Pt->next)
      count++;
   return count;
}
```