厦门大学《数据结构》期末试题·答案

考试日期：2006.1（zch）　　信息学院自律督导部

一、试设计算法在 O(n)时间内将数组 A[1..n]划分为左右两个部分，使得左边的所有元素奇数，右边的所有元素均为偶数，要求所使用的辅助存储空间大小为 O(1)。

解：该题算法的主要思路如下：

（1）设置两个指针 i 和 j，其中 i=1，j=n。

（2）当 i<j 时作如下循环：

　　i 不断自加从左往右找到第一个偶数

　　j 不断自减从右往左找到第一个奇数

　　A[i]与 A[j]交换

（3）算法结束

```
Adjust(int A[1..n])
{
    int i=1, j=n;
    while (i<j ){
        while (A[i] % 2 != 0 && i<=n) i++;
        while (A[i] % 2 = =0 && j>=1) j--;
        if (i>n || j<1) break;
        if (i<j) A[i] ←→ A[j];
    }
}
```

算法的时间复杂性为 O(n)，辅助存储空间为 O(1)。

二、写一个算法将一带头结点的单链表逆转，要求利用原表结点空间，不允许申请新的结点空间。

解：

方法一：建立一个新的单链表，其中的结点从原表得来，即每个原表中得到一个结点，就要将此结点插入新链表中。由于要将表逆转，原表的头结点成为新链表的头结点，每次从原表中得到一个结点，此结点插在头结点之后，作为新链表的第一个结点。

```
void InverLinkedList( LinkList &L){
    LNode *p, *s;
    P=L→next;
    L→next=NULL;
    while (p) {
        s=p;            //p 为待逆置链表头指针
        p=p→next;  //从 p 所指链表中删除第一个结点
        s→next=L→next;
        L→next=s;        //将 s 结点插入到逆置表的表头
    }
}
```

方法二：在遍历原表的时候将各结点的指针逆转，从原表的第一个结点开始，表头结点的指针在最后修改成指向原表的最后一个结点。

```
void InvertLinkedList( LinkList &L) {
    LNode *p, *q;
    S=L→next;
    if (s) {
        q=NULL;
        p=s;
        while (p) {
            p=p→next;
            s→next=q;
            q=s;
            s=p;
        }
        L→next=q;
    }
}
```

三、设 T 是一棵具有 n 个节点的二叉树，若给定二叉树 T 的先序序列和中序序列，并假设 T 的先序序列和中序序列分别放在数组 PreOrder[1..n]和 InOrder[1..n]中，设计一个构造二叉树 T 的链式存储结构的算法。以下为结点类型：

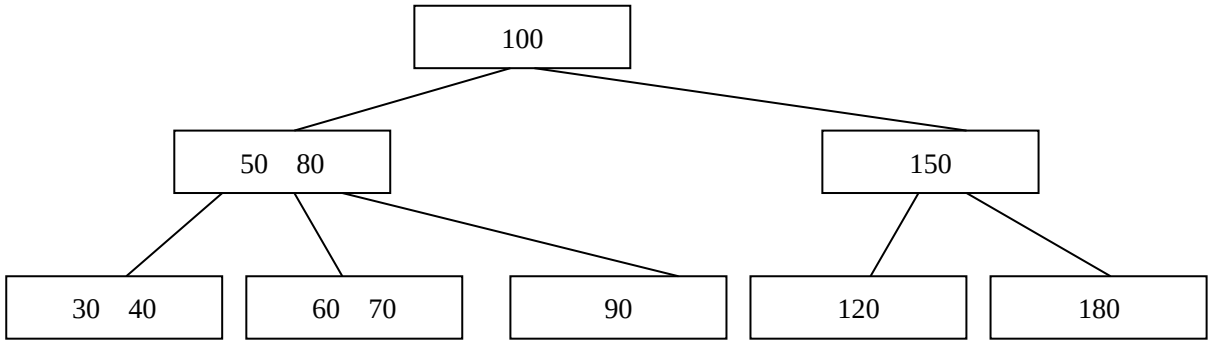
```
typedef struct BiTNode{
    TElemType data;
    Struct BiTNode *lchild, *rchild;
} BiTNode, *BiTree;
```

解：设 T 的先序序列和中序序列分别放在数组 PreOrder[1..n]和 InOrder[1..n]中，根据先序遍历的特点可知，PreOrder[1]为根节点，设中序序列中 InOrder[i]=PreOrder[1]，则在 InOrder[i]的左面的 InOrder[1..i-1]应为二叉树的左子树，而 InOrder[i+1..n]为根的右子树。相对应的是，在先序序列中根的左子树应为 PreOrder[2..i]，而右子树为 PreOrder[i+1..n]。而左子树的根为 PreOrder[2]，右子树的根为 PreOrder[i+1]。依次递归，用同样的方法可以确定子树的左子树和右子树，从而逐步确定整个二叉树。

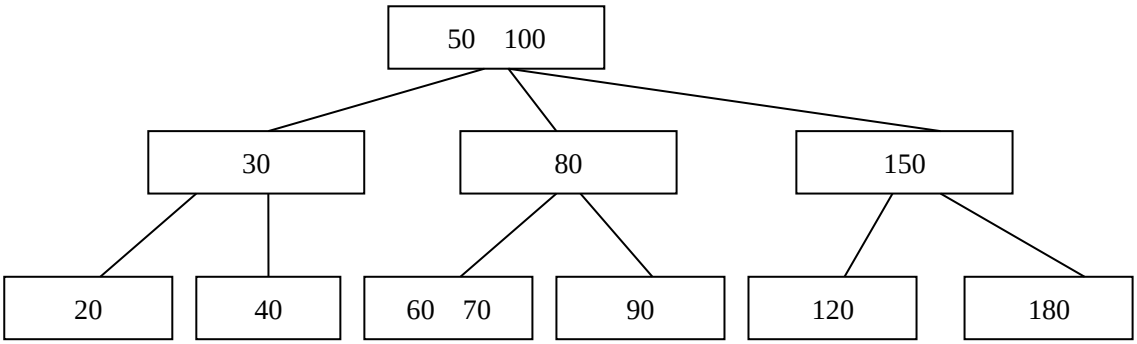
```
typedef struct BiTNode{
    TElemType data;
    Struct BiTNode *lchild, *rchild;
} BiTNode, *BiTree;

void Create(BiTree T, TElemType PreOrder[], int i1, int j1, TElemType InOrder[], int i2, int j2)
{
    int i=i2;
    if (i1<=j1){
        T=(BiTree) malloc(sizeof(BiNode));
        T→data=PreOrder[i1];
        T→lchild=NULL；
        T→rchild=NULL;
        while (InOrder[i]!=PreOrder[i1]) i++;
```

```
        Create(T→lchild, PreOrder, i1+1, i-i2+i1, InOrder, i2, i-1);
        Create(T→rchild, PreOrder, i-i2+i1+1, j1, InOrder, i+1, j2);
    }
    else T=NULL;
}
```

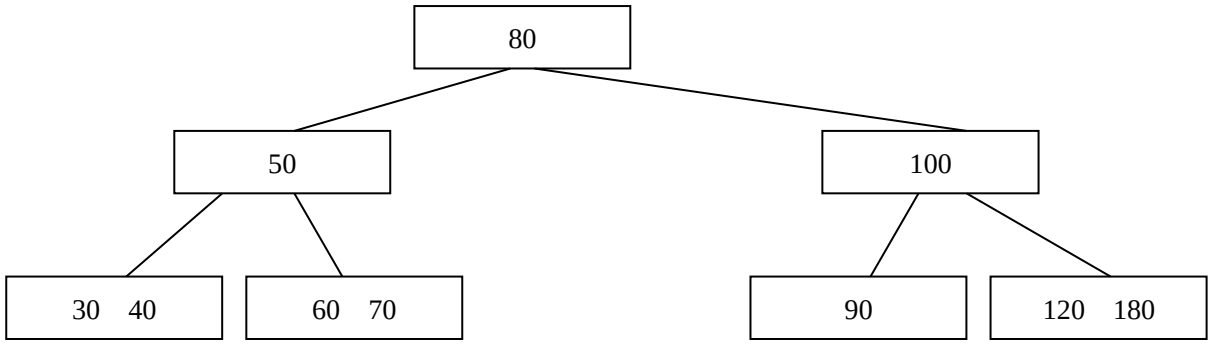
四、设有 3 阶 B－树，如下图所示，分别画出在该树插入关键字 20 和在原树删除关键字 150 得到的 B－树。



解：插入 20 后的 B－树为：



删除 150 后的 B－树为：



五、已知待散列存储的关键字序列为（4,15,38,49,33,60,27,71），哈希函数为 H(key)=key MOD 11，哈希表 HT 的长度为 11，采用二次探测再散列法解决冲突，试构造此哈希表，并求出在等概率情况下查找成功的平均查找长度。

解：哈希表为

0	1	2	3	4	5	6	7	8	9	10
33	60		27	4	15	38	71		49	

平均查找长度为：(1+2+2+4+1+5+6+7)/8 =3.5。

六、有一种简单的排序算法，叫做计数排序。这种排序算法对一个待排序的表进行排序，并将排序结果存放到另一个新的表中。必须注意的是，表中所有待排序的关键字互不相同。计数排序算法针对表中的每个记录，扫描待排序的表一趟，统计表中有多少个记录的关键字比该记录的关键字要小。假设针对某一个记录，统计出的计算值为 c，那么这个记录在新的有序表中的合适的存放位置为 c+1。

（1） 编写实现计数排序的算法；

（2） 分析该算法的时间复杂性。

解：（1）假设数据结构如下：

```
#define MAXSIZE    20
typedef int KeyType；
typedef struct {
    KeyType    key;
    InfoType otherinfo;
} RedType;
typedef struct {
    RedType      r [MAXSIZE + 1 ] ; // r[0]  空或作哨兵
    int                    length;
} SqList;

void CountSort(SqList L1, SqList L2)
{//把 L1 计数排序后，结果放在 L2
     int i, j, n, count;
     n=L1.length;
     L2.length=L1.length;
     for (i=1; i<=n; i++){
          count=0;
          for (j=1; j<=n; j++)
               if (L1.r[j]<L1.r[i]) count++;
          L2.r[count+1]=L1.r[i];
     }
}
```

（2）基本操作是关键字比较操作和记录移动操作。其中关键字比较操作为 $O(n^2)$，记录移动操作为 $O(n)$。因此，总的时间复杂性为 $O(n^2)$。

七、请谈谈学习《数据结构》课程的心得体会，并以某个算法为例谈谈对该算法的理解。

解：无标准答案