武汉大学计算机学院

2006 年－2007 学年第二学期 “数据结构”考试试题 （A）

姓名 _____ 学号(序号)_____ 班号 _____

要求：所有的题目的解答均写在答题纸上 （每张答题纸上要写清楚姓名、 班号和学号），需写清楚题目的序号。每张答题纸都要写上姓名和序号。

一、单项选择题（每小题 2 分，共 20 分）

1. 在存储数据时，通常不仅要存储各数据元素的值，而且还要存储 ____。

A. 数据的处理方法　　B. 数据元素的类型

C. 数据元素之间的关系　　D. 数据的存储方法

2. 下述函数中对应的渐进时间复杂度（ n 为问题规模）最小是 ____。

A.$T1(n)=n\log_2 n+5000n$　　B.$T2(n)=n^2-8000n$

C.$T3(n)=n^{\log_2 n}-6000n$　　D.$T4(n)=1000n\log_2 n+7000\log_2 n$

3. 设线性表有 n 个元素，以下操作中，_____在顺序表上实现比在链表上实现效率更高。

A. 输出第 i(1 ≤i ≤n)个元素值

B. 交换第 1 个元素与第 2 个元素的值

C. 顺序输出这 n 个元素的值

D. 输出与给定值 x 相等的元素在线性表中的序号

4. 设 n 个元素进栈序列是 p_1，p_2，p_3，…，p_n，其输出序列是 1，2，3，…，n，若 $p_3=3$，则 p_1 的值 ____。

A. 可能是 2　　B. 一定是 2

C. 不可能是 1　　D. 一定是 1

5. 以下各种存储结构中，最适合用作链队的链表是 ____。

A. 带队首指针和队尾指针的循环单链表　　B. 带队首指针和队尾指针的非循环单链表

C. 只带队首指针的非循环单链表　　D. 只带队首指针的循环单链表

6. 对于链串 s（长度为 n，每个结点存储一个字符），查找元素值为 ch 的算法的时间复杂度为 ____。

A.O(1)　　B.O(n)

C.$O(n^2)$　　D. 以上都不对

7. 设二维数组 A[6][10]，每个数组元素占用 4 个存储单元，若按行优先顺序存放的数组元素 a[3][5] 的存储地址为 1000，则 a[0][0] 的存储地址是 ____。

A.872　　B.860

C.868　　D.864

8. 一个具有 1025 个结点的二叉树的高 h 为____。

A.11　　B.10

C.11～1025　　D.12～1024

9. 一棵二叉树的后序遍历序列为 DABEC，中序遍历序列为 DEBAC，则先序遍历序列为 ____。

A.ACBED B.DECAB

C.DEABC D.CEDBA

10. 对图 1 所示的无向图，从顶点 1 开始进行深度优先遍历；可得到顶点访问序列 ____。

A.1 2 4 3 5 7 6 B.1 2 4 3 5 6 7

C.1 2 4 5 6 3 7 D.1 2 3 4 5 7 6

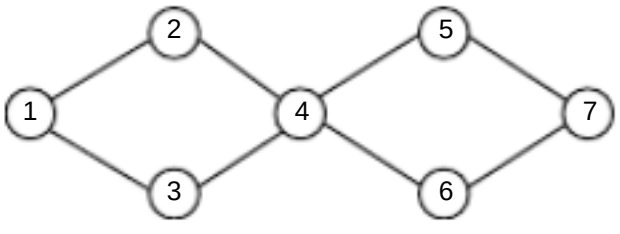

图 1 一个无向图

二、填空题（每题 2 分，共 10 分）

1. 顺序队和链队的区别仅在于 ____的不同。

2. 在有 n 个顶点的有向图中，每个顶点的度最大可达 _____。

3. 对有 18 个元素的有序表 R[1..18] 进行二分查找，则查找 R[3] 的比较序列的下标为 _____。

4. 对含有 n 元素的关键字序列进行直接选择排序时，所需进行的关键字之间的比较次数为 _____。

5. 已知关键字序列为 {2,7,4,3,1,9,10,5,6,8}，采用堆排序法对该序列作升序排序时，构造的初始堆（大根堆）是 _____。（不用画出堆，只需写出初始堆的序列）

三、问答题（共 40 分）

1. 一棵完全二叉树上有 1001 个结点，其中叶结点的个数是多少？（需写出推导过程，8 分）

2. 给出如下各种情况下求任意一个顶点的度的过程（只需文字描述）：（8 分）

（1）含 n 个顶点的无向图采用邻接矩阵存储；

（2）含 n 个顶点的无向图采用邻接表存储；

（3）含 n 个顶点的有向图采用邻接矩阵存储；

（4）含 n 个顶点的有向图采用邻接表存储。

3. 将整数序列 {4,5,7,2,1,3,6} 中的数依次插入到一棵空的平衡二叉树中，试构造相应的平衡二叉树。（要求画出每个元素插入过程，若需调整，还需给出调整后的结果，并指出是什么类型的调整，12 分）

4. 当实现插入直接排序过程中，假设 R[0..i-1] 为有序区，R[i..n-1] 为无序区，现要将 R[i] 插入到有序区中，可以用二分查找来确定 R[i] 在有序区中的可能插入位置，这样做能否改善直接插入排序算法的时间复杂度 ?为什么 ?（8 分）

5. 简述外排序的两个阶段。（4 分）

四、算法设计题　（共 30 分）

1. 设计一个算法　delminnode(LinkList *&L)　，在带头结点的单链表　L 中删除所有结点值最小的结点（可能有多个结点值最小的结点）　　。（15 分）

2. 假设二叉树采用二叉链存储结构存储,　　设计一个算法　copy(BTNode *b,BTNode *&t)　，由二叉树　b 复制成另一棵二叉树　t。（15 分）

参　考　答　案

一、单项选择题（每小题　2 分，共 20 分）

1. C　　2. D　　3. A　　4. A　　5. B

6. B　　7. B　　8. C　　9. D　　10. A

二、填空题　（每题 2 分，共 10 分）

1. 存储方法或存储结构。
2. 2(n-1)。
3. 9、4、2、3
4. n(n-1)/2。
5. 10,8,9,6,7,2,4,5,3,1。（序列不全对不给分）

三、问答题（共　40 分）

1. 答：二叉树中度为　1 的结点个数只能是　1 或 0。设 $n_1=1$，$n=n_0+n_1+n_2=n_0+n_2+1=1001$，由性质　1 可知 $n_0=n_2+1$，由两式可求　$n_0=500.5$，不成立；设　$n_1=0$，$n=n_0+n_1+n_2=n_0+n_2=1001$，由性质　1 可知 $n_0=n_2+1$，由两式可求　$n_0=501$。本题答案为：　501。

评分标准：只给出结果给　　3 分，推导过程占　5 分。

2. 答：对于邻接矩阵表示的无向图，顶点　　i 的度等于第　i 行中元素等于　1 的个；对于邻接矩阵表示的有向图，顶点　　i 的出度等于第　i 行中元素等于　1 的个数；入度等于第　i 列中元素等于　1 的个数；度数等于它们之和。

对于邻接矩阵表示的无向图，顶点　　i 的出度等于　g->adjlist[i]　为头结点的单链表中结点的个数；入度需要遍历各顶点的边表，若　　g->adjlist[k]　为头结点的单链表中存在顶点编号为 i 的结点，则顶点　i 的入度增　1；度数等于它们之和。

评分标准：有向图、无向图两种存储方式各占　　4 分。

3. 建立平衡二叉树过程如图　　2 所示 (图中加阴影的结点表示要调整的结点　　)。

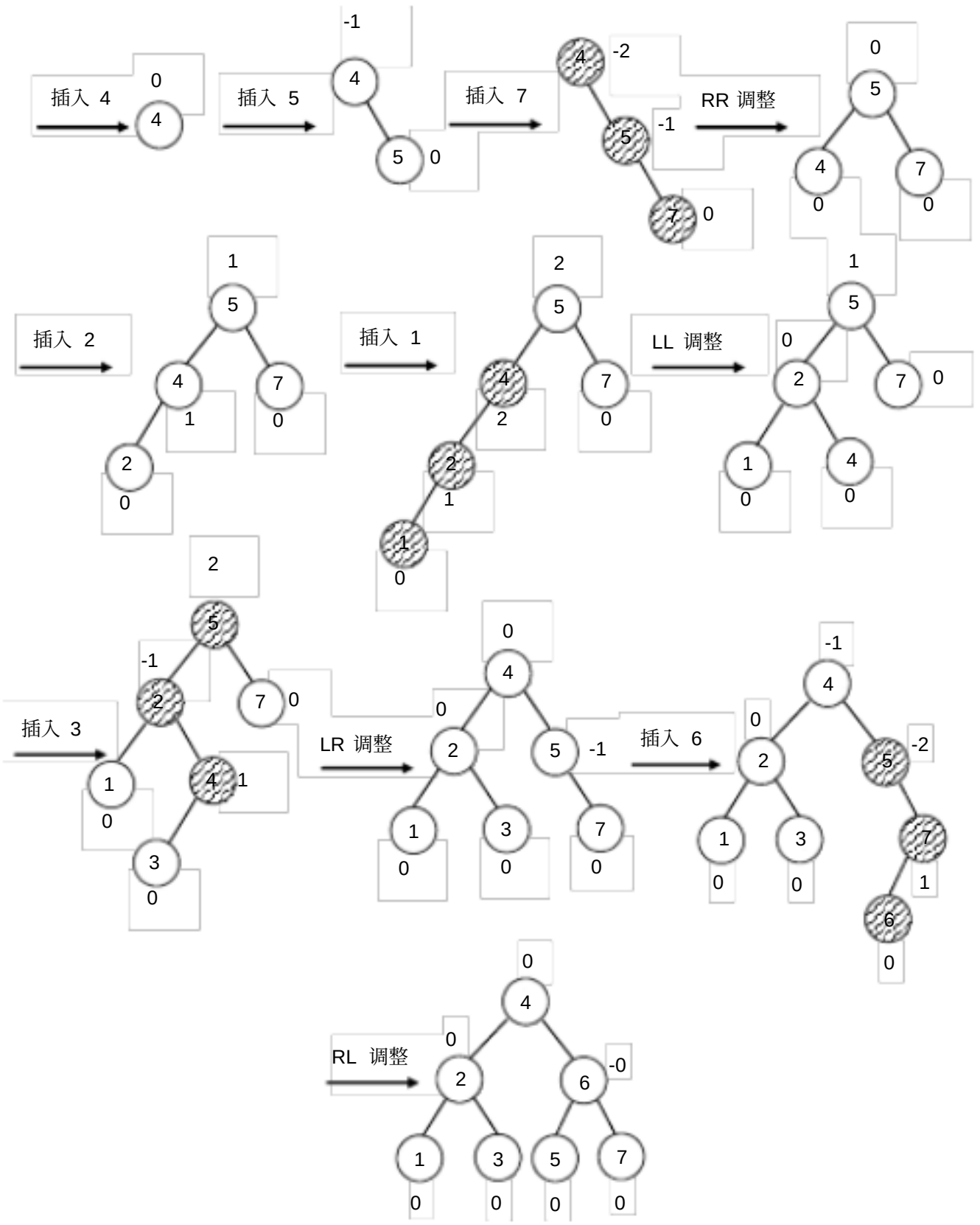

图 2 构造平衡二叉树过程

评分标准：每次调整占 1 分。

4. 答：不能。因为在这里，二分查找只减少了关键字间的比较次数，而记录的移动次数不变，时间的复杂度仍为 $O(n^2)$。

评分标准：答对“不能”占 3 分，说明理由占 5 分。

5. 答：生成初始归并段（或顺串），采用多路平衡归并方法进行归并。

四、算法设计题 （共 30 分）

1. 解：用 p 从头至尾扫描单链表，pre 指向 *p 结点的前驱，用 minp 保存值最小的结点指针，minpre 指向 *minp 结点的前驱。一面扫描，一面比较，将最小值的结点放到 *minp 中。算法如下：

```
void delminnode(LinkList *&L)
{
```

```
LinkList *pre=L,*p=pre->next,*minp=p,*minpre=pre;
    ElemType mindata=p->data;
    while (p!=NULL && p->data<mindata)
    {       mindata=p->data;
            p=p->next;
    }
    p=pre->next;
while (p!=NULL)
{
            if (p->data==mindata)
            {       pre->next=p->next;
                    free(p);
                }
                pre=pre->next;
                p=pre->next;
        }
}
```

评分标准：根据算法的正确性评分，不考虑算法的时间复杂度。

2. 解：递归算法如下：

```
void copy(BTNode *b,BTNode *&t)
{
        BTNode *l,*r;
        if (b==NULL) t=NULL;
        else
        {
                t=(BTNode *)malloc(sizeof(BTNode));
                copy(b->lchild,l);
                copy(b->rchild,r);
                t->lchild=l;
                t->rchild=r;
        }
}
```

评分标准：根据算法的正确性评分，不考虑算法的时间复杂度。