



# 厦门大学《数据结构》期末试题·答案

考试日期：2007·1 (A)

信息学院自律督导部



## 一、(本题 10 分)

- (1) 简述线性表的两种存储结构的主要优缺点及各自适用的场合。
- (2) 在折半查找和表插入排序中，记录分别应使用哪种存储结构，并用一句话简述理由。

答：(1) 顺序存储是按索引（如数组下标）来存取数据元素，优点是可实现快速的随机存取，缺点是插入与删除操作将引起元素移动，降低效率。对于链式存储，元素存储采取动态分配，利用率高。缺点是须增设指针域，存储数据元素不如顺序存储方便。优点是插入与删除操作简单，只须修改指针域。

(2) 在折半查找中，记录使用顺序存储，可以快速实现中点的定位；在表插入排序中，记录使用静态链表，可以减少移动记录的操作。

二、(本题 15 分) 在带头结点的非空线性链表中，试设计一算法，将链表中数据域值最小的那个结点移到链表的最前面，其余各结点的顺序保持不变。要求：不得额外申请新的链结点。

解：程序如下：

```
typedef struct node {
    int data;
    struct node * next;
}Node,*LinkList;
void MinFirst(LinkList L)
{Node *p,*q,*ptrmin;
    if(L->next == NULL) return; //空表
    ptrmin = L; //ptrmin 指向当前最小结点的前一个结点
    p = L->next; //p 指向当前结点的前一个结点
    while(p->next!=NULL) {
        if( p->next->data < ptrmin->next->data ) ptrmin = p;
        p = p->next;
    }
    //q 指向最小结点，并从链表中删除
    q = ptrmin->next; ptrmin->next = q->next;
    q->next = L->next; L->next = q; //q 指向的最小结点插入到链表头
}
```

三、(本题 15 分) 编写函数判断一棵二叉树是否不含有度为 1 的结点，若任何结点的度都不为 1，则返回 TRUE，否则返回 FALSE。二叉树采用标准的二叉链表实现。注意：函数的代码要有注释。

解：结点与二叉树的数据结构如下：

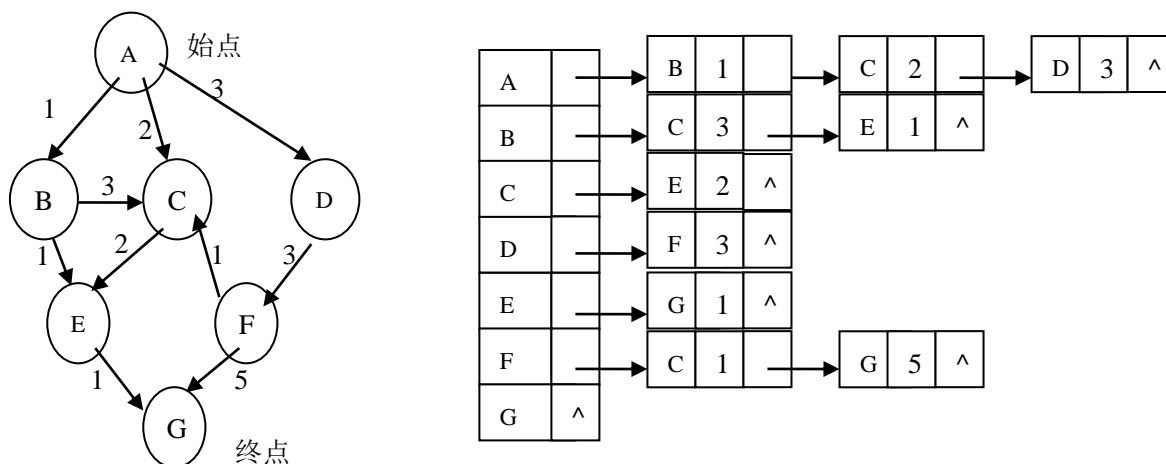
```

typedef struct BiTNode {
    TElemType data;
    struct BiTNode * lchild, * rchild; //左右孩子指针
} BiTNode, * BiTree;

bool NoSingleBranchTree(BiTree T)
{
    if( T==NULL) return true;
    if( T->lchild == NULL && T->rchild !=NULL) return false;
    if( T->lchild != NULL && T->rchild ==NULL) return false;
    return NoSingleBranchTree(T->lchild)&&NoSingleBranchTree(T->rchild);
}

```

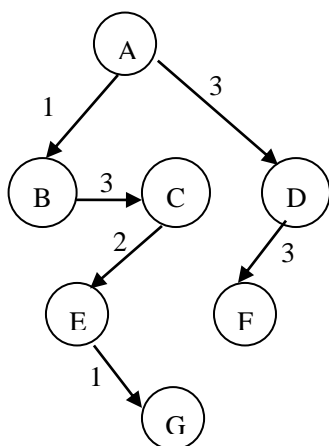
四、（本题 15 分）某带权有向图及其邻接表如下：



- (1) 写出深度优先搜索结点访问序列；（邻接边的顺序按照邻接表链表顺序）
- (2) 画出深度优先生成树；
- (3) 将该图作为 AOE 网络，写出顶点 C 的最早发生时间及活动 FC 的最晚开始时间。

解：

- (1) 深度优先搜索顺序是：A, B, C, E, G, D, F
- (2) 深度优先生成树如下图所示。

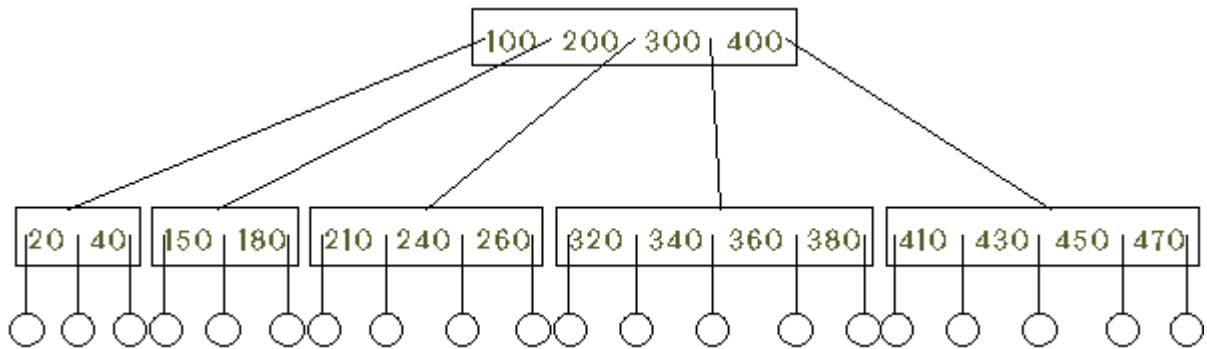


(3) 求解过程如下:

$ee(A)=0$     $ee(D)=3$     $ee(F)=ee(D)+3=6$     $ee(B)=1$ ;  
 $ee(C)=\max\{ ee(A)+2, ee(B)+3, ee(F)+1\}=7$   
 $ee(E)=\max\{ ee(B)+1, ee(C)+2 \}=9$     $ee(G)=\max\{ ee(E)+1, ee(F)+5 \}=11$   
 $le(G)=11$     $le(E)=le(G)-1=10$     $le(C)=le(E)-2=8$   
 $le(B)=\min\{le(E)-1, le(C)-3\}=5$   
 $le(F)=\min\{ le(G)-5, le(C)-1 \}=6$     $le(D)=le(F)-3=3$   
 $le(A)=\min\{ le(B)-1, le(C)-2, le(D)-3 \}=0$   
 则  $l(FC)=le(C)-1=7$   
 所以, C 的最早发生时间为 7, 活动 FC 的最晚开始时间为 7。

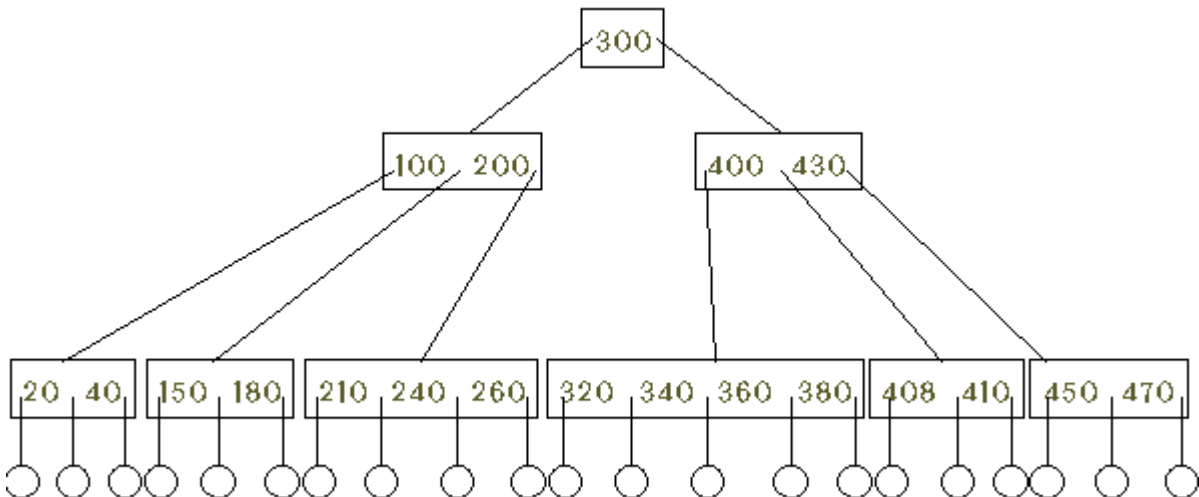
五、(本题 15 分)

(1) 请画出往如下图的 5 阶 B-树中插入一个关键字 408 后得到的 B-树, 以及再删除关键字 180 后得到的 B-树;

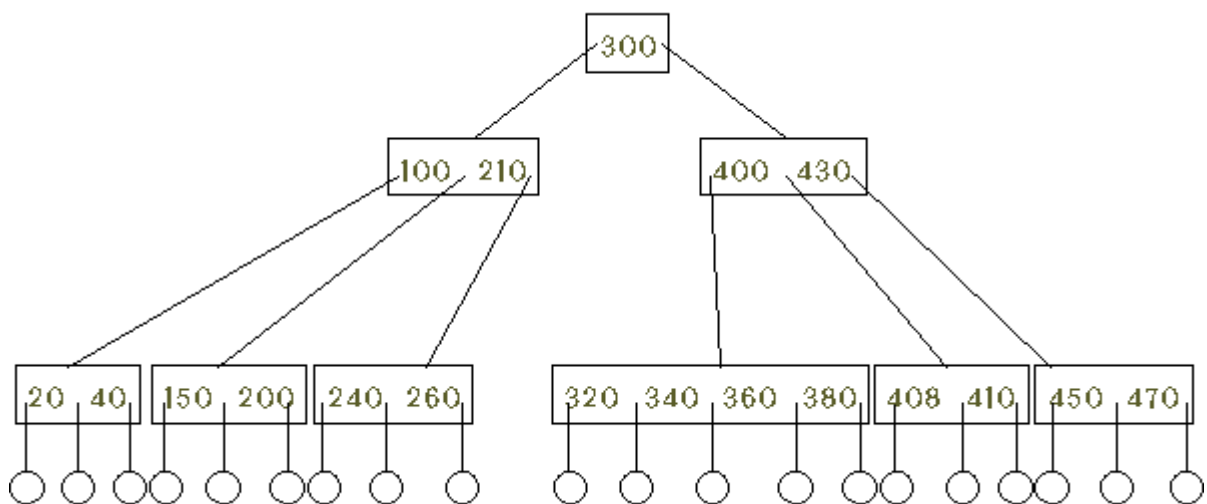


(2) 包括  $n$  个关键字的  $m$  阶 B-树在一次检索中最多涉及多少个结点? (要求写出推导过程)

解: (1) 插入一个关键字 408 后得到的 B-树



再删除关键字 180 后得到的 B-树



(2) 根据 B-树的定义，第一层至少有 1 个结点；第二层至少有 2 个结点；由于除根之外的每个非终端结点至少有  $\lceil m/2 \rceil$  棵子树，则第三层至少有  $2\lceil m/2 \rceil$  个结点；……；依次类推，第  $L+1$  层至少有  $2(\lceil m/2 \rceil)^{L-1}$  个结点。而  $L+1$  层的结点为叶子结点。若  $m$  阶 B-树种具有  $n$  个关键字，则叶子结点即查找不成功的结点为  $n+1$ ，由此有：

$$N+1 \geq 2(\lceil m/2 \rceil)^{L-1}$$

$$\text{反之, } l \leq \log_{\lceil m/2 \rceil} \left( \frac{N+1}{2} \right) + 1$$

这就是说，在含有  $n$  个关键字的 B-树上进行查找时，从根结点到关键字所在结点的路径上涉及的结点数不超过  $\log_{\lceil m/2 \rceil} \left( \frac{N+1}{2} \right) + 1$ 。

六、(本题 15 分) 以关键字序列 (29, 18, 25, 47, 58, 12, 51, 10) 为例，执行以下排序算法，写出每一趟结束时的关键字状态：

(1) 增量序列为 5, 3, 1 的希尔排序 (2) 快速排序 (3) 堆排序。

解：(1)

第 1 趟： 12, 18, 10, 29, 47, 58, 51, 25

第 2 趟： 12, 18, 10, 29, 25, 58, 51, 47

第 3 趟： 10, 12, 18, 25, 29, 47, 58, 51

(2)

第 1 趟： 10, 18, 25, 12, 29, 58, 51, 47

第 2 趟： 10, 18, 25, 12, 29, 47, 51, 58

第 3 趟： 10, 12, 18, 25, 29, 47, 51, 58

第 4 趟： 10, 12, 18, 25, 29, 47, 51, 58

(3)

第 1 趟： 58, 47, 51, 29, 18, 12, 25, 10

第 2 趟： 51, 47, 25, 29, 18, 12, 10, (58)

第 3 趟: 47, 29, 25, 10, 18, 12, (51, 58)  
 第 4 趟: 29, 18, 25, 10, 12, (47, 51, 58)  
 第 5 趟: 25, 18, 12, 10, (29, 47, 51, 58)  
 第 6 趟: 18, 10, 12, (25, 29, 47, 51, 58)  
 第 7 趟: 12, 10, (18, 25, 29, 47, 51, 58)  
 第 8 趟: 10, (12, 18, 25, 29, 47, 51, 58)

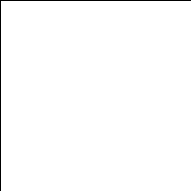
七、(本题 15 分)二路归并排序是从  $n$  个长度为 1 的有序子序列开始的, 一种改进方法是先对待排序序列扫描一遍, 并把它划分为若干个长度最大的有序子序列, 然后从这些有序子序列开始进行两两归并。例如, 若待排序序列为 (15, 18, 2, 26, 43, 92, 89, 25, 28, 30, 36, 12), 先扫描一遍划分为 (15, 18), (2, 26, 43, 92), (89), (25, 28, 30, 36), (12) 这 5 个有序子序列, 然后从这 5 个子序列开始两两归并。试设计恰当的存储表示方法, 并在此基础上实现该算法。

解: 设  $n$  个待排序元素存放在不带头结点的单链表中, 每个结点存放一个元素, 头指针为  $r$ 。只修改结点中的链指针而不移动结点中的元素值, 排序后  $r$  仍指向结果链表的第一个结点。

```
void Merger(LinkList La, LinkList Lb, LinkList &Lc) {
    if (La && Lb) {
        if (La->data <= Lb->data) {
            Lc = La; pa = La->next; pb = Lb;
        }
        else {
            Lc = Lb; pa = La; pb = Lb->next;
        }
        pc = Lc;
        while (pa && pb)
            if (pa->data <= pb->data) {
                pc->next = pa; pc = pa; pa = pa->next;
            }
            else {
                pc->next = pb; pc = pb; pb = pb->next;
            }
            if (pa)
                pc->next = pa;
            if (pb)
                pc->next = pb;
    }
}
```

```
void MergeSort(LinkList &r) {
    //初始化
    InitQueue(Q);
    if (!r)
        return;
```

```
    //划分出若干个最大有序子序列, 并依次将这些子序列的头指针进队列
    s = r;
    EnQueue(Q, r);
    while (s) {
        t = s->next;
```



```
while (t && s->data<=t->data) {  
    s=t; t=t->next; }  
if (t) {  
    s->next = NULL; s=t; EnQueue(Q, s); }  
}
```

//从队首取出两个有序子序列，归并它们，并将得到的新的有序子序列的头指针进队列，直到从队列只能剩 1 个有序子序列为止。

```
while (!QueueEmpty(Q)) {  
    DeQueue(Q, r);  
    if (QueueEmpty(Q))  
        break;  
    DeQueue(Q, s);  
    Merge(r, s, t);  
    EnQueue(Q, t);  
}  
}
```