

# 数据结构强化课考试

注1：本次考试时间共120分钟（408考试中，平均每个大题可分配20分钟左右的做题时间，因此该试卷共包含6个数据结构大题）

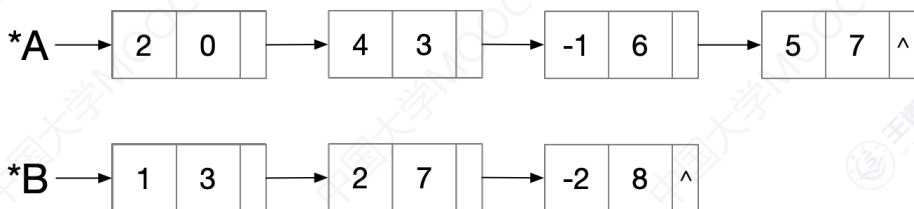
注2：本试卷的个别题目难度略高于真题

## 算法题

一、数学上的一元多项式  $P(x) = p_0 x^0 + p_1 x^1 + p_2 x^2 + \dots + p_n x^n$ ，可以用一个单链表来存储，结点的数据结构定义如下：

```
typedef struct Node {  
    float co;    //系数  
    int ex;      //指数  
    struct Node *next;    //指向下一个结点  
} *Polynomial;
```

其中，co 表示每一项的“系数”，ex 表示每一项的“指数”，next 为指向下一个结点的指针。我们规定：“系数”为0的项无需存储；各个项在链表中按“指数”递增存放；单链表没有头结点。例如，多项式  $A(x) = 2 + 4x^3 - x^6 + 5x^7$  和 多项式  $B(x) = x^3 + 2x^7 - 2x^8$  可表示为：



现要求设计一个尽可能高效的算法，实现两个多项式的加法，并返回相加之后的结果。要求：

- 1) 给出算法的基本设计思想。
- 2) 根据设计思想，采用C或C++语言描述算法，关键之处给出注释。
- 3) 说明你所设计算法的时间复杂度和空间复杂度。

### 【参考答案】

1) 算法思想：

每个多项式的各个项，按照指数递增的次序存放在单链表中。两个多项式相加，可以将相加的结果合并到一个链表中。

可设置两个指针 p、q，分别用于遍历 A(x)、B(x) 的各个结点

若当前 p、q 所指两个结点的“指数 ex”相等，则，将 p、q 的系数相加，并让 p 指向 A(x) 的下一个结点，q 指向 B(x) 的下一个结点；

若当前 p 的“指数 ex”更小，则将 q 所指结点插入到 p 结点之前，保证指数更小的项在单链表中更靠前，再让 q 指向 B(x) 的下一个结点；



若当前 p 的“指数 ex”更大，则将 p 指针指向 A(x) 的下一个结点即可（两个多项式相加的结果保存在原来的单链表 A(x) 中）。

2) 算法代码如下：

```
//a和b两个多项式相加，相加的结果合并到a
int Addition (Polynomial &a, Polynomial &b){
    if (a == NULL || b == NULL)
        return 0;          //两个多项式不能为空

    struct Node *p=a, *q=b;      //p、q 两个指针遍历a、b两个单链表
    struct Node *pPre=NULL;      //pPre 用于记录 p 的前驱
    while (p!=NULL && q!= NULL) {
        if (p->ex < b->ex) {      //p结点的指数更小
            pPre = p;
            p = p->next;
        } else if (p->ex > q->ex) {    //q结点的指数更小，将q插入到p之前
            //单链表中，实现前插操作不方便，可以先后插，再交换数据
            struct Node * s = q;
            q = q->next;
            s->next = p->next;
            p->next = s;
            //交换两个结点的数据
            int coTemp = s->co;
            int exTemp = s->ex;
            s->co = p->co;
            s->ex = p->ex;
            p->co = coTemp;
            p->ex = exTemp;
            //p指向下一个待合并结点
            pPre = p;
            p=p->next;
        } else {                  //p和q所指结点的指数相同，则统一合并到p
            p->co += q->co;
            pPre = p;
            p = p->next;
            struct Node * s = q;
            q = q->next;
            free(s);
        }
    }
    if (p==NULL) { //p==NULL，说明多项式b中还有一些项没有合并
        pPre->next = q;
    }
}
```

3) 若两个多项式的长度分别是 m、n，则

时间复杂度 =  $O(m+n)$

空间复杂度 =  $O(1)$

二、已知一棵非空二叉树 T 高度为 h，结点总数为 n，采用二叉链表存储结构，结点的数据结构定义如下：

```
//二叉树的结点（链式存储）
typedef struct BiTNode{
    char data;                //数据域
    struct BiTNode *lchild,*rchild; //左、右孩子指针
}BiTNode,*BiTree;
```

请设计一个算法，求树 T 的宽度（即具有结点数最多的那一层的结点个数），要求：

- 1) 给出算法的基本设计思想。
- 2) 根据设计思想，采用C或C++语言描述算法，关键之处给出注释。

【参考答案】

1) 算法思想：

由题目已知二叉树高度为 H，因此可设置用于统计的数组 width[H]，其中，width[i] 表示第 i 层共有几个结点（不妨规定层数从 0 开始）；

使用二叉树前序遍历的思想，遍历的同时，判断结点所处层数 i，令 width[i]++；

遍历完所有结点后，找到 width 数组中最大的值，即为二叉树的宽度。

2) 算法代码如下：

```
//先序遍历，同时统计各层结点总数
void PreOrder(BiTree T, int level, int width[]){
    if (T == NULL) return;
    width[level]++; //根据当前结点所处层数，累加该层结点总数
    PreOrder(T->lchild, level + 1, width); //遍历左子树
    PreOrder(T->rchild, level + 1, width); //遍历右子树
}

//求树的宽度
int treeWidth (BiTree T) {
    int width[H]; //定义一个数组，用于统计各层的结点总数
    for (int i=0; i<H; i++)
        width[i]=0; //数组初始化

    PreOrder(T, 0, width); //前序遍历二叉树，同时统计各层结点总数

    int maxWidth = 0; //找到最大宽度
    for (int i=0; i<H; i++){
        if(width[i]>maxWidth)
            maxWidth =width[i];
    }
}
```

```
return maxWidth;           //返回树的宽度  
}
```

注：本题为王道书 5.3.3 大题 14 题，王道书用层序遍历实现，略麻烦。平时训练可以用层序遍历，但考试中只要题目没有特殊要求，就尽量用前/中/后序递归遍历实现。

三、一个长度为 $n$ 的升序整数序列 $S$ 中，只有常数 $K$ （ $K$ 的值已知）出现了若干次，其他数最多都只出现一次。试设计一个在时间和空间两方面都尽可能高效的算法，返回  $K$  出现的次数。

- (1) 给出算法的基本设计思想。
- (2) 根据设计思想，采用C或C++语言描述算法，关键之处给出注释。
- (3) 说明你所设计算法的时间复杂度和空间复杂度。

【参考答案】

次优解：枚举，遍历数组统计 $k$ 出现的次数

最优解：折半查找两次

次优解：枚举，遍历数组统计 $k$ 出现的次数

- 1) 算法思想：循环遍历数组，统计数组中 $k$ 出现的次数。
- 2) 算法如下：

```
int ans(int A[], n, k){  
    int m=0;           //m统计k出现的次数  
    for (int i=0; i<n; i++){  
        if (A[i]==k)  
            m++;  
    }  
    return m;  
}
```

- 3) 时间复杂度： $O(n)$                       空间复杂度： $O(1)$

最优解：折半查找两次

- 1) 算法思想：要求出 $k$ 出现的次数，可以折半查找 $k-1$ 和 $k+1$ 的位置，他们之间就是 $k$ 出现的次数。 $k-1$ 和 $k+1$ 最多只会出现一次，通过折半查找 $k-1$ 和 $k+1$ 的位置，然后将下标相减就得到 $k$ 出现的次数。
- 2) 算法如下：

```
int Binary_Search(int A[], int L, int R, int x){  
    int mid;  
    while (L<R){           //如果L>R则范围错误  
        mid=(L+R)/2;       //mid取中间数，向下取整  
        if (x<=A[mid])
```

```
        R=mid;
    else L=mid+1;           //更新查找范围
}
return L;                  //返回数组下标L
}

//ans函数返回数组s中，常数k出现的次数
int ans(int S[], int n, int k){
    int L=Binary_Search(int A[], 0, n-1, k-1);
    if (A[L]<k)
        L++;                //L是k出现的最小下标
    int R=Binary_Search(int A[], 0, n-1, k+1);
    if (A[L]>k)
        R--;                //R是k出现的最大下标
    return R-L+1;
}
```

3) 时间复杂度:  $O(\log n)$

空间复杂度:  $O(1)$

## 应用题

四、请回答以下问题:

- (1) 队列在顺序存储时的“假溢出”现象指什么?
- (2) 简述一种可行的假溢出的解决方法。
- (3) 若用数组 $q[1...m]$ 表示队列, 队列头指针 $front$ 、尾指针 $rear$ 的初值均为1, 基于(2)中的方法, 如何求队列的当前长度? 如何判定队空? 如何判定队满?

【北京邮电大学803 2018年】

考点: 队列的应用

【参考答案】

(1)

对于顺序存储的队列, 执行入队和出队操作时, 头、尾指针只增大不减小, 致使出队元素的空间无法被重新利用。因此, 尽管队列中实际元素个数可能远小于数组大小, 但可能由于尾指针已超出数组下标的界限而不能执行入队操作。该现象称为“假溢出”。

(咸鱼注: 有的简答题, 如果你觉得文字表述不够清晰, 可能会导致老师扣你分, 那么可以主动配上示例图, 比如加上后边这段...当然, 如果你觉得文字表达已经OK了, 完全可以不画图)

如下图所示, 是数组大小为5的顺序队列, 队尾指针超出了数组范围, 尽管此时仍有未使用的空间, 但无法入队, 此时发生“假溢出”。



Q.rear →

Q.front →

a5
a4
a3

(2)

解决假溢出的方法，可以采用循环队列。入队操作时，若队列不满，则将新元素插入队尾指针 rear 所指位置，并令  $rear = (rear+1)\%m$ ；出队操作时，若队列不空，则令队头指针 front 所指元素出队，并令  $front = (front+1)\%m$ 。其中，m为队列总长度。

(3)

队空条件：front==rear

队满条件：front==(rear+1)%m

队列长度：length = (rear-front+m)%m

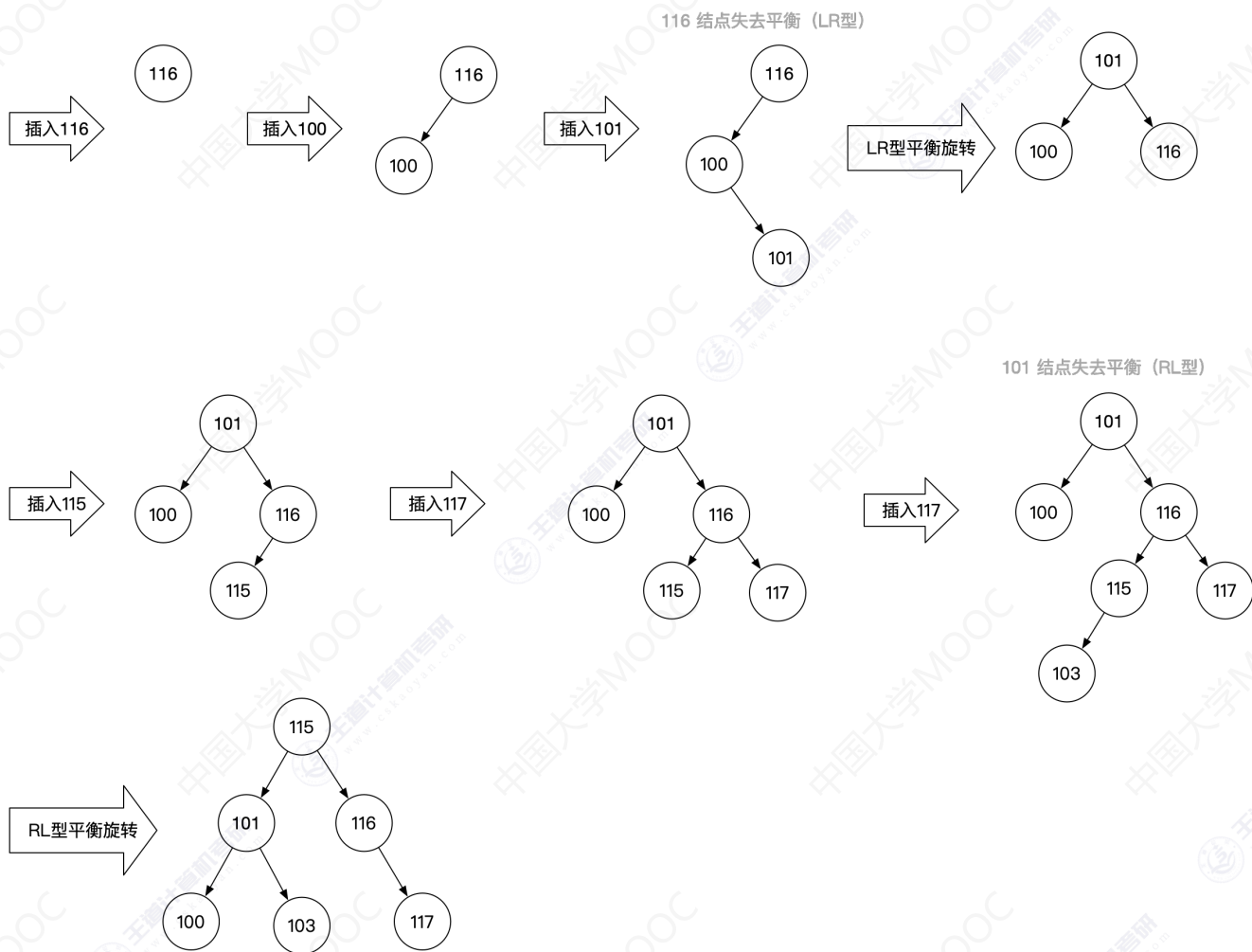
五、将关键字序列 {116, 100, 101, 115, 117, 103} 依次插入到初始为空的平衡二叉树（AVL树），给出每插入一个关键字后的平衡树，并说明其中可能包含的平衡调整步骤（即：先说明是哪个结点失去平衡，然后说明做了什么平衡处理）；然后分别给出前序、中序和后序遍历该二叉树的输出结果。【中国科学院大学863 2019年】

考点：平衡二叉树

【参考答案】

各关键字的插入过程如下所示：





对该二叉树的遍历结果如下

前序遍历: 115, 101, 100, 103, 116, 117

中序遍历: 100, 101, 103, 115, 116, 117

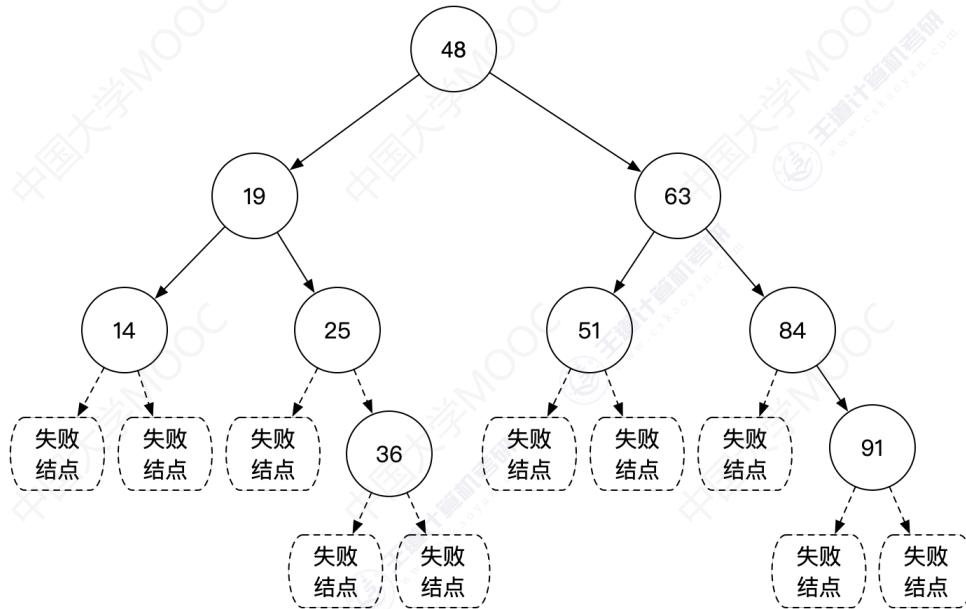
后序遍历: 100, 103, 101, 117, 116, 115

六、设一组有序的记录关键字序列为 (14,19,25,36,48,51,63,84,91)，运用二分法进行查找，请给出二分查找的判定树，以及查找关键字 84 时的比较次数，并计算出查找成功时的平均查找长度。【华中科技大学834 2019年】

考点: 查找算法的分析与应用 (二分查找)

【参考答案】

二分查找的判断树如下:



查找成功时的平均查找长度  $ASL_{成功} = (1 + 2 \times 2 + 3 \times 4 + 4 \times 2) / 9 = 25 / 9$