第五章 数组&广义表

讨论题:

1、广义表具有哪些重要特性？ 2、若在矩阵 Amn*中存在一个元素 ai-1,j-1是第 i 行元素中最小值又是第 j 列元素中 的最大值，则称此元素为该矩阵的一个马鞍点。假设以二维数组存储矩阵 Amn*， 试设计一个求该矩阵所有马鞍点的算法，并分析你的算法在最坏情况下的时间复 杂度。 3、广义表 GL=（a1,a2,…an），其中 ak(k=1,2, …n)或是单个数据元素（原子）， 或仍然是个广义表。编写一个过程或函数计算一个广义表的所有原子节点数据域 之和，例如对广义表（3，（2,4,5），（6,3）），数据域之和为 23。 4、设A和B均为下三角矩阵，每一个都有n行。因此在下三角区域中各有n(n+1)/2 个元素。另设有一个 二维数组 C，它有 n 行 n+1 列。试设计一个方案，将两个 矩阵 A 和 B 中的下三角区域元素存放于同一个 C 中。要求将 A 的下三角区域中的 元素存放于 C 的下三角区域中，B 的下三角区域中的元素转置后存放于 C 的上三 角区域中。并给出计算 A 的矩阵元素 aij和 B 的矩阵元素 bij在 C 中的存放位置下标的公式。

课后题

5.17（1）、（3），

5.17③ 已知顺序表 L 含有 n 个整数，试分别以函数形式写出下列运算的递归算法：

(1) 求表中的最大整数；
(2) 求表中的最小整数；
(3) 求表中 n 个整数之和；
(4) 求表中 n 个整数之积；
(5) 求表中 n 个整数的平均值。

```
int a[];
int max_L(int i,int x){
    if(i==0)return a[i];
    return max(a[x],max_L(i-1));
}

print(max_L(n))
```

```
int a[]
int sum_L(int i){
    if(i==0)return a[i];
    return a[i]+sum_L(i-1);
}
print(sum_L(n))
```

5.19、5.30、5.32、5.33、5.37、5.38

5.19④ 若矩阵 $A_{m\times n}$ 中的某个元素 a_{ij} 是第 i 行中的最小值，同时又是第 j 列中的最大值，则称此元素为该矩阵中的一个马鞍点。假设以二维数组存储矩阵 $A_{m\times n}$，试编写求出矩阵中所有马鞍点的算法，并分析你的算法在最坏情况下的时间复杂度。

```
int minn[M],maxx[N];
void MinMax(int A[M][N]) //M行中最小，N列中最大
{
    int i,j;
    bool have=false;
    for(i=0; i<M; i++) //求出每行最小数，存在minn[0,,,M-1]中
    {
        minn[i]=A[i][0];
        for(j=1; j<N; j++)
            if(minn[i]>A[i][j])
                minn[i]=A[i][j];
    }
    for(j=0; j<N; j++) //求出每列最大数，存在maxx[0,,,N-1]中
    {
        maxx[j]=A[0][j];
        for(i=1; i<M; i++)
            if(maxx[j]<A[i][j])
                maxx[j]=A[i][j];
    }
    for(i=0; i<M; i++)
        for(j=0; j<N; j++)
            if(minn[i]==maxx[j]) //找到马鞍点
            {
                printf("A[%d][%d]=%d",i,j,A[i][j]);
                have=true;
            }
    if(!have)
        printf("没有马鞍点\n");
}
```

30.

5.30③ 试按表头、表尾的分析方法重写求广义表的深度的递归算法。

```
int GListDepth(GList L)
{
    int m, n;

    if(!L)                              //空表深度为1
        return 1;

    if(L->tag==Atom)                    //原子深度为0
        return 0;

    m = GListDepth(L->Union.ptr.hp) + 1; //表头深度
    n = GListDepth(L->Union.ptr.tp);     //表尾深度

    return m>n ? m : n;
}
```

31

5.31③ 试按教科书 5.5 节图 5.10 所示结点结构编写复制广义表的递归算法。

```

Status Equal(GList A, GList B)
{
    if(!A && !B)                                //两个空表相等
        return OK;

    if(A && B)                                  //两个表均不为空
    {
        if(A->tag==B->tag)                      //元素类型相同
        {
            if(A->tag==Atom)                    //原子结点
            {
                if(A->Union.atom==B->Union.atom)
                    return OK;
            }
            else                                //表结点
            {
                if(Equal(A->Union.ptr.hp, B->Union.ptr.hp))
                {
                    if(Equal(A->Union.ptr.tp, B->Union.ptr.tp))
```

```
                        return OK;
                }
            }
        }

    return ERROR;
}
```

33

5.33④ 试编写递归算法，输出广义表中所有原子项及其所在层次。

```
void Print(GList L, int d)
{
    int i = d;                                  //d的初值赋值为0

    if(L)
    {
        if(L->tag==Atom)
            printf("%c -> 第%d层\n", L->Union.atom, i);

        if(L->tag==List)                        //表头指针指向表的话层数增一
        {
            Print(L->Union.ptr.hp, i+1);
            Print(L->Union.ptr.tp, i);
        }
    }
}
```

37

5.37⑤ 试编写递归算法，删除广义表中所有值等于 x 的原子项。

```

void ClearGList_GL_H_T(GList *L)
{
    GList p, q;

    if(*L)
    {
        if((*L)->tag==Atom)
        {
```

```
            free(*L);                                   //删除原子结点
            *L = NULL;
        }
        else                                            //删除表结点
        {
            p = (*L)->Union.ptr.hp;
            q = (*L)->Union.ptr.tp;
            free(*L);
            *L = NULL;
            ClearGList_GL_H_T(&p);
            ClearGList_GL_H_T(&q);
        }
    }
}

void DeleteX(GList *L, AtomType x)
{
    GList h, p;

    if(*L && (*L)->tag==List)
    {
        h = (*L)->Union.ptr.hp;
        if(h)
        {
            if(h->tag==List)
            {
                DeleteX(&((*L)->Union.ptr.hp), x);
                DeleteX(&((*L)->Union.ptr.tp), x);
            }
            else
            {
                if(h->Union.atom==x)
                {
                    p = *L;
                    *L = (*L)->Union.ptr.tp;
                    p->Union.ptr.tp = NULL;
                    ClearGList_GL_H_T(&p);
                    DeleteX(L, x);
                }
                else
                    DeleteX(&((*L)->Union.ptr.tp), x);
            }
        }
        else
        {
            if((*L)->Union.ptr.tp)
                DeleteX(&((*L)->Union.ptr.tp), x);
        }
    }
```

```
}

```

38

5. 38④ 试编写算法，依次从左至右输出广义表中第 l 层的原子项。

```
void PrintL(GList L, int d, int l)
{
    int i = d;                              //d的初值赋值为0

    if(L && l>=i)
    {
        if(L->tag==Atom)
        {
            if(l==i)                        //层数符合
                printf("%c ", L->Union.atom);
        }
        else                                //表头指针指向表的话层数增一
        {
            PrintL(L->Union.ptr.hp, i+1, l);
            PrintL(L->Union.ptr.tp, i, l);
        }
    }
}
```