**1．试设计一个算法：如果在矩阵A中存在一个元素aij（0≤i≤m-1，0≤j≤n-1），该元素是第i行元素中最小值且又是第j列元素中最大值，则称此元素为该矩阵的一个马鞍点。假设以二维数组常规存储矩阵A，求解该矩阵的所有马鞍点。**

解答：

void MaAn(int a[ ][ ], int m, int n) {

for (i = 0; i < n; i++) {

min = a[i][0]; //min为第i行中的最小值

k = 0;

for (j = 1; j < m; j++){

if (a[i][j] < min) {

min = a[i][j];

k = j;

} // a[i][k]为第i行最小值

}

for (j = 0; j < n; j++) {

if (a[j][k] > min)

break;

if (j == n)

cout<< min; // 输出鞍点

}

}

}

分析算法，外层for循环共执行n次，内层第一个for循环执行m次，第二个for循环最坏情况下执行n次，所以，最坏情况下的时间复杂度为Ｏ(n2+mn)。

**2．试设计一个算法：求解一个广义表所拥有的原子结点个数。**

解答：

typedef enum{ATOM,LIST} ElemTag;

typedef struct GLNode{

ElemTag tag; //区分原子节点和子表

union{

AtomType data;

struct {

struct GLNode \*hp，\*tp; //表结点的表头表尾指针

}ptr;

};

}GLNode;

typedef GLNode \*GList;

//以上为定义广义表的类型

int count(GList GL){

int m,n; //m存储下一个结点的递归调用的值，n存储子表递归调用的值

if(! GL) return 0;

else{

if(GL->tag==ATOM) n=1;

else //递归求解原子结点个数

n=count(GL->ptr.hp);

if(GL->ptr.tp !=NULL)

m=count(GL->ptr.hp);

else m=0; //没有下一个结点

return (n+m);

}

}

1. **魔方阵，又叫幻方阵，在我国古代称为“纵横图”。它是在一个n×n的矩阵中填入1到n2的数字（n为奇数），使得每一行、每一列、每条对角线的累加和都相等。例如下图所示就是一个3阶魔方阵。要求：（1）试设计存储结构表示魔方阵；（2）试设计算法完成任意n阶魔方阵的填数。**

解答：

（1）数据结构设计 int m[100][100];

将魔方阵储存在二维数组中，实验中可直接应用二维数组。

（2）算法如下

void mofang(int m){ //输入阶数为m（奇数）

M=100;

a[M][M]={0}; //先令所有元素都为0

while(!((m!=0)&&(m<M)&&(m%2!=0))){

printf("请重新输入:");

scanf("%d",&m);

}

x=0; y=m/2; a[x][y]=1; //由1开始填数，将1放在第0行的中间位置；

for(k=2;k<=m\*m;k++){ //将魔方阵想象成上下左右相接， 每次往左上角走一步

x=(x-1+m)%m; //左上角超出上边边界，则在最下边相对应的位置填入下一个数字

y=(y-1+m)%m; //左上角超出左边边界，则在最右边相对应的位置填入下一个数字

if(a[x][y]==0){

a[x][y]=k;

}

else{  //若按上述方法找到位置已填入数据，则在同一列下一行填入下个数字

x=(x+2)%m;

y=(y+1)%m;

a[x][y]=k;

}

}

for(x=0;x<m;x++){ //显示出n阶魔方阵

for(y=0;y<m;y++)

printf("%d\t",a[x][y]);

printf("\n");

}

}