1.试将稀疏矩阵A的非零元素表示成三元组的形式和十字链表的形式

略，因该题很简单的。

2.试编写一个以三元组形式输出十字链表表示的稀疏矩阵中非零元素的算法。

注意：十字链表添加了两个链域：一个是链接同一行下一个非零元结点的 right 域，另一个是链接同一列下一个非零元结点的 down 域。每个非零元既是某个行链表中的一个结点，又是某个列链表中的一个结点，整个矩阵构成了一个十字交叉的链表，故称为十字链表，以两个分别存放行链表的头指针和列链表的头指针的一维数组表示之。

算法：稀疏矩阵的十字链表存储表示:

```
typedef struct OLNode {
   int i,j; //  该非零元的行和列下标
   ElemType e; //  非零元素值
   OLNode *right,*down; //  该非零元所在行表和列表的后继链域
}OLNode, *OLink;

typedef struct {
   OLink *rhead,*chead; //  行和列链表头指针向量基址
   int mu,nu,tu; //  稀疏矩阵的行数、列数和非零元个数
}CrossList;
```

```
void OutCSM(CrossList M, void(*Out3)(int, int, int))

/*  用函数Out3,依次以三元组格式输出十字链表表示的矩阵  */

{       int i;    OLink p;

        for(i=0;i<=M.mu;i++)

        {

          if(M.rhead[i])

          for(p=M.rhead[i];p;p=p->right)

          Out3(i,p->j,p->e);

          }

 }
```

3. 分别实现以三元组存储、十字链表存储稀疏矩阵的加法操作A=A+B的算法，并分析各自的时间复杂性。

算法：这里应该加上一个条件，即假设三元组顺序表A或十字链表A的空间足够大

1）　三元组存储：

```
void TSMatrix_Addto(TSMatrix &A,TSMatrix B)//将三元组矩阵B加到A上
{
  for(i=1;i<=A.tu;i++)
    A.data[MAXSIZE-A.tu+i]=A.data[i];/把A的所有元素都移到尾部以腾出位置
  pa=MAXSIZE-A.tu+1;pb=1;pc=1;
  for(x=1;x<=A.mu;x++) //对矩阵的每一行进行加法
  {
    while(A.data[pa].i<x) pa++;
    while(B.data[pb].i<x) pb++;
    while(A.data[pa].i==x&&B.data[pb].i==x)//行列值都相等的元素
    {
      if(A.data[pa].j==B.data[pb].j)
      {
        ne=A.data[pa].e+B.data[pb].e;
```

```
        if(ne) //和不为0
        {
          A.data[pc].i=x;
          A.data[pc].j=A.data[pa].j;
          A.data[pc].e=ne;
          pa++;pb++;pc++;
        }
      }//if
      else if(A.data[pa].j>B.data[pb].j)
      {
        A.data[pc].i=x;
        A.data[pc].j=B.data[pb].j;
        A.data[pc].e=B.data[pb].e;
        pb++;pc++;
      }
      else
      {
        A.data[pc].i=x;
        A.data[pc].j=A.data[pa].j;
        A.data[pc].e=A.data[pa].e
        pa++;pc++;
      }
    }//while
    while(A.data[pa]==x) //插入A中剩余的元素(第x行)
    {
      A.data[pc].i=x;
      A.data[pc].j=A.data[pa].j;
      A.data[pc].e=A.data[pa].e
      pa++;pc++;
    }
    while(B.data[pb]==x) //插入B中剩余的元素(第x行)
    {
      A.data[pc].i=x;
      A.data[pc].j=B.data[pb].j;
      A.data[pc].e=B.data[pb].e;
      pb++;pc++;
    }
  }//for
  A.tu=pc;
  for(i=A.tu;i<MAXSIZE;i++) A.data[i]={0,0,0}; //清除原来的A中记录
}//TSMatrix_Addto
```

时间复杂度为O(m,n)，其中m和n分别为A,B矩阵中非零元的数目

2)十字链表存储：

算法思想：以rowTail colTail数据空间换定位链尾的时间，我想对于特别大的矩阵有用，要注意的是有的指针一定要初始化，否则后面用 !=NULL将不能得到正确的判断结果，这里程序有点长，注意点都在旁边加了注释，应该有更简洁的代码，欢迎发我邮箱 ssxbily@163.com ,一起讨论。

算法：

```
#include <stdio.h>
#include <malloc.h>

//数据结点定义
typedef struct OLNode{
int i,j;
int value;
struct OLNode *right,*down;
}OLNode;

//十字链表类型定义
typedef struct{
OLNode *rhead,*chead;
int mu,nu,tu;//稀疏矩阵行数、列数和非零元个数
}CrossList;

//创建一个 CrossList，包含行列链头，数据要初始化
int createList(CrossList* list,int m,int n){
//data check
if(list == NULL){
   return -1;
}
if(m<1 || n<1){
   return -1;
}

//initialize list
list->mu = m;
list->nu = n;
list->tu = 0;
list->rhead = (OLNode*)malloc(sizeof(OLNode));
```

```
list->chead = (OLNode*)malloc(sizeof(OLNode));
list->rhead->i = -1;
list->rhead->j = -1;
list->rhead->right = NULL;//指针一定要显示初始化，否则后面用 !=NULL 将不能得到正确的判断结果
list->rhead->down = NULL;
list->chead->i = -1;//该有的值就应该写上，后面调试才好好发现错误在哪里！
list->chead->j = -1;
list->chead->right = NULL;
list->chead->down = NULL;

//initialize rhead

OLNode *nodeAdd = list->rhead;
for(int i=0; i<m; i++){
   nodeAdd->down = (OLNode*)malloc(sizeof(OLNode));
   nodeAdd = nodeAdd->down;
   nodeAdd->i = i+1;//对行的标号是从第 1 行、第 2 行……起的，但 i 从 0 开始，故此处为 i+1，下同
   nodeAdd->j = -1;
   nodeAdd->right = NULL;
   nodeAdd->down = NULL;
}
//initialize chead
nodeAdd = list->chead;
for(int i=0; i<n; i++){
   nodeAdd->right = (OLNode*)malloc(sizeof(OLNode));
   nodeAdd = nodeAdd->right;
   nodeAdd->i = -1;//指示列链表头
   nodeAdd->j = i+1;
   nodeAdd->right = NULL;
   nodeAdd->down = NULL;
}

return 0;
}

//销毁链表
void destroyList(CrossList* list){
//data check
if(list == NULL){
   return;
}
//销毁横链链头
OLNode *node = list->rhead;
```

```
OLNode *p = node;
while(p != NULL){
   node = node->down;
   free(p);
   p = node;
}

//销毁纵链链头
node = list->chead;
p = node;
while(p != NULL){
   node = node->right;
   free(p);
   p = node;
}

//销毁链表
free(list);
}

//从文本文件读入数据,失败返回-1，成功返回 0
int readList(FILE* fp,CrossList* list,int m, int n){
//data check
if(list == NULL){
   return -1;
}
if(m<1 || n<1){
   return -1;
}

//创建链表
createList(list,m,n);

//生成行列链链尾数组
OLNode ** rowTail = (OLNode**)malloc(m*sizeof(OLNode*));//指示每行链表表尾
OLNode ** colTail = (OLNode**)malloc(n*sizeof(OLNode*));//指示每行链表表尾
OLNode* op = list->rhead->down;
int i = 0;
while(op != NULL){
   rowTail[i] = op;
   op = op->down;
   i++;//i++提出来写吧，不要显示自己级别高
}
op = list->chead->right;
i = 0;
```

```
while(op != NULL){
   colTail[i] = op;
   op = op->right;
   i++;
}

int value;
//read list data
for(int i=0; i<m; i++){
   for(int j=0; j<n; j++){
    if(fscanf(fp,"%d",&value) != 1){//fscanf()以空白字符作为分隔，因此空格符、换行符是分隔符，不用考虑换行问题！fscanf()返回读取的数据个数
     printf("输入数据有误！\n");
     free(rowTail);
     free(colTail);
     return -1;
    }
    if(value != 0){//非零元素加入链表
     //新结点初始化
     OLNode* node = (OLNode*)malloc(sizeof(OLNode));
     node->right = NULL;
     node->down = NULL;
     node->i = i+1;
     node->j = j+1;
     node->value = value;
     list->tu ++;

     rowTail[i]->right = node;//以 rowTail colTail 数据空间换定位链尾的时间，我想对于特别大的矩阵有用吧
     rowTail[i] = node;
     colTail[j]->down = node;
     colTail[j] = node;
    }
   }
}

free(rowTail);//记得回收哈
free(colTail);
return 0;
}

//将两链表相加：A=A+B 用到第三个链表
CrossList* addListT(CrossList* listA,CrossList* listB){
//data check
if((listA == NULL) || (listB == NULL)){
```

```
    return NULL;
}
int m = listA->mu;
int n = listA->nu;
if((m != listB->mu) || (n != listB->nu)){
    printf("链表维数不符合要求! \n");
    return NULL;
}

//创建结果数组并初始化
CrossList* resultCrossList = (CrossList*)malloc(sizeof(CrossList));//用于暂存结果
createList(resultCrossList,m,n);
resultCrossList->tu = listA->tu + listB->tu;

//生成结果链表的行列链链尾数组
OLNode ** resultRowTail = (OLNode**)malloc(m*sizeof(OLNode*));//指示每行链表表尾
OLNode ** resultColTail = (OLNode**)malloc(n*sizeof(OLNode*));//指示每行链表表尾
OLNode* rclNode = resultCrossList->rhead->down;
int i = 0;
while(rclNode != NULL){
    resultRowTail[i] = rclNode;
    rclNode = rclNode->down;
    i++;
}
rclNode = resultCrossList->chead->right;
i = 0;
while(rclNode != NULL){
    resultColTail[i] = rclNode;
    rclNode = rclNode->right;
    i++;
}

OLNode* rah = listA->rhead;
OLNode* rbh = listB->rhead;
for(int i=0; i<m; i++){//对每行链表相加
    rah = rah->down;//指示到当前行
    rbh = rbh->down;
    OLNode* p = rah->right;
    OLNode* q = rbh->right;
    while((p != NULL) && (q != NULL)){//两行均未完
     int c1 = p->j;
     int c2 = q->j;
```

```
    OLNode* node = (OLNode*)malloc(sizeof(OLNode));
    node->right = NULL;//可别忘记了这里的显示初始化
    node->down = NULL;
    if(c1<c2){
     node->i = i+1;
     node->j = c1;
     node->value = p->value;
     p = p->right;
    }else if(c1>c2){
     node->i = i+1;
     node->j = c2;
     node->value = q->value;
     q = q->right;
    }else{//将结点数据相加
     int re = p->value + q->value;
     if(re != 0){//数据相加不为零
      node->i = i+1;
      node->j = c1;//c1 c2 都是一样的哈
      node->value = re;
      p = p->right;
      q = q->right;
      resultCrossList->tu --;//重复计数，减一
     }else{
      free(node);//非此时，更待何时
      p = p->right;
      q = q->right;
      resultCrossList->tu -= 2;//重复两次计数，减二
      continue;//避免执行//end while 前面的四语句
     }
    }//end if
    //这个函数里以下四个语句比较一致，故提到 if 语句外面来写哈，但在本程序另外一个函数里就不行了哈
    resultRowTail[i]->right = node;
    resultRowTail[i] = node;
    resultColTail[i]->down = node;
    resultColTail[i] = node;
   }//end while
   while(p != NULL){//链表 A 这一行未完的
    OLNode* node = (OLNode*)malloc(sizeof(OLNode));
    node->i = p->i;
    node->j = p->j;
    node->value = p->value;
    node->right = NULL;
    node->down = NULL;
```

```
    resultRowTail[i]->right = node;
    resultRowTail[i] = node;
    resultColTail[i]->down = node;
    resultColTail[i] = node;
    p = p->right;
   }
   while(q != NULL){//链表 B 这一行未完的
    OLNode* node = (OLNode*)malloc(sizeof(OLNode));
    node->i = q->i;
    node->j = q->j;
    node->value = q->value;
    node->right = NULL;
    node->down = NULL;

    resultRowTail[i]->right = node;
    resultRowTail[i] = node;
    resultColTail[i]->down = node;
    resultColTail[i] = node;
    q = q->right;
   }
}//end for

free(resultRowTail);
free(resultColTail);

return resultCrossList;
}

//将两链表相加：A=A+B 直接将 B 加到 A 上
int addList(CrossList* listA,CrossList* listB){
//data check
if((listA == NULL) || (listB == NULL)){
   return -1;
}
int m = listA->mu;
int n = listA->nu;
if((m != listB->mu) || (n != listB->nu)){
   printf("链表维数不符合要求！\n");
   return -1;
}

//生成结果链表的行列链链尾数组
listA->tu += listB->tu;
OLNode ** resultRowTail = (OLNode**)malloc(m*sizeof(OLNode*));//指示每行链表
表尾
```

```
OLNode ** resultColTail = (OLNode**)malloc(n*sizeof(OLNode*));//指示每行链表表尾
OLNode* rclNode = listA->rhead->down;
int i = 0;
while(rclNode != NULL){
   resultRowTail[i] = rclNode;
   rclNode = rclNode->down;
   i++;
}
rclNode = listA->chead->right;
i = 0;
while(rclNode != NULL){
   resultColTail[i] = rclNode;
   rclNode = rclNode->right;
   i++;
}

OLNode* rah = listA->rhead;
OLNode* rbh = listB->rhead;
for(int i=0; i<m; i++){//对每行链表相加
   rah = rah->down;//指示到当前行
   rbh = rbh->down;
   OLNode* p = rah->right;
   OLNode* q = rbh->right;
   while((p != NULL) && (q != NULL)){//两行均未完
    int c1 = p->j;
    int c2 = q->j;
    if(c1<c2){
     resultRowTail[i]->right = p;
     resultRowTail[i] = p;
     resultColTail[i]->down = p;
     resultColTail[i] = p;
     p = p->right;
    }else if(c1>c2){
     OLNode* node = (OLNode*)malloc(sizeof(OLNode));
     node->right = NULL;//可别忘记了这里的显示初始化
     node->down = NULL;
     node->i = i+1;
     node->j = c2;
     node->value = q->value;

     resultRowTail[i]->right = node;
     resultRowTail[i] = node;
     resultColTail[i]->down = node;
```

```
      resultColTail[i] = node;
      q = q->right;
     }else{//将结点数据相加
      int re = p->value + q->value;
      if(re != 0){//数据相加不为零
       p->value = re;

       resultRowTail[i]->right = p;
       resultRowTail[i] = p;
       resultColTail[i]->down = p;
       resultColTail[i] = p;
       p = p->right;
       q = q->right;
       listA->tu --;//重复计数，减一
      }else{
       p = p->right;
       q = q->right;
       listA->tu -= 2;//重复两次计数，减二
      }
     }//end if
    }//end while
    if(p != NULL){//链表 A 这一行未完的
     resultRowTail[i]->right = p;
     resultColTail[i]->down = p;
    }
    while(q != NULL){//链表 B 这一行未完的
     OLNode* node = (OLNode*)malloc(sizeof(OLNode));
     node->i = q->i;
     node->j = q->j;
     node->value = q->value;
     node->right = NULL;
     node->down = NULL;

     resultRowTail[i]->right = node;
     resultRowTail[i] = node;
     resultColTail[i]->down = node;
     resultColTail[i] = node;
     q = q->right;
    }
}//end for

free(resultRowTail);
free(resultColTail);
```

```
return 0;
}

//将链表内的数据显示出来
void showList(CrossList* list){
//data check
if(list == NULL){
   return;
}

OLNode* row = list->rhead->down;//location row list header
OLNode* node;
while(row != NULL){
   node = row->right;
   while(node != NULL){
    printf("(%d,%d,%d)\t",node->i,node->j,node->value);
    node = node->right;
   }
   printf("\n");
   row = row->down;
}
}

int main(){//请严格将申请的内存NULL化
FILE* fp = fopen("data.txt","r");
if(fp == NULL){
   printf("Can't open file!\n");
   getchar();//getchar()便于看到结果，而不是程序功能一实现就结束了。
   return -1;
}

//get CrossList info
int m,n;
fscanf(fp,"%d%d",&m,&n);

//read list data
CrossList* listA = (CrossList*)malloc(sizeof(CrossList));
CrossList* listB = (CrossList*)malloc(sizeof(CrossList));
CrossList* result;
if((readList(fp,listA,m,n) == 0) && (readList(fp,listB,m,n) == 0)){//数据读
入正确
   printf("链表A的内容为：\n");
   showList(listA);
   printf("链表B的内容为：\n");
   showList(listB);
```

```
    result = addListT(listA,listB);
    if(result != NULL){
     printf("使用第三方链表相加之后，结果链表为：\n");
     showList(result);
    }
    if(addList(listA,listB) == 0){
     printf("不使用第三方链表相加之后，结果链表为：\n");
     showList(listA);
    }
}

destroyList(listA);
destroyList(listB);
destroyList(result);
fclose(fp);

getchar();
}
```

4.设GetHead为CAR，GetTail为CDR，求下列广义表的运算结果：

(1)CAR((x, y, z))

(2)CDR((x, y, z))

(3)CAR(((a, b), (x, y)))

(4)CDR(((a, b), (x, y)))

(5)CAR(CDR(((a, b), (x, y))))

(6)CDR(CDR(((a, b), (x, y))))

(7)CAR(CDR(CDR(((a, b), (x, y)))))

答案：(1) x; (2) (y,z) ; (3)(a,b) ; (4)((x,y)) ;

(5) (x,y); (6) ( ) ; (7)( )

5. 设广义表A=((a,b),(a,(b,(c))),d,((a,c),d)) 求表头、表尾。

答案：表头(a,b)，表尾(a,(b,(c))),d,((a,c),d)。