完成日期：2016-10-30

* **问题描述**

假设稀疏矩阵A和B均以三元组表作为存储结构，试写出矩阵相加(相乘—选做)的算法，另设三元组表C存放结果矩阵。

矩阵相加测试实例：

0 7 0 8 0 2 0 1 0 0

A= 4 0 0 0 5 B= 4 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 6 0 6 0 0 -6 0

处理过程提示：

* + 输入稀疏矩阵A和B
  + 检测A和B能否相加
  + 矩阵相加运算
  + 打印运算结果
* **算法思想：**

要求用三元组表存储，即存储非零元素的位置和数值。先判断AB矩阵的形态（行列）是否相同，判断是否可以相加。通过判断AB两三元组表内各非零元素位置来决定直接传入C还是相加再传入，分为A直接传入，B直接传入，相加传入，跳过 四种结果。

* **设计描述**

1）**数据存储结构类型的定义**；

typedef struct{

int i,j;//该非零元的行下标和列下标

int e;//数值

}Triple,\*TriplePtr;

typedef struct{

Triple data[MAXSIZE+1];//非零元三元组表，data[0]未用

int mu,nu,tu;//矩阵的行数、列数和非零元个数

}TSMatrix,\*TSPtr;

2）**程序的处理流程以及各模块之间的调用关系；**

int initial(int a,int b,TSPtr A)//初始化

int enter(int u,int a,int b,int n,TSPtr A)//输入存储

int addMa(TSPtr A,TSPtr B,TSPtr C)//加法

void output(TSPtr Q)//输出

main函数先读取行列，初始化A，之后调用enter函数以三元组表方式存储矩阵，同理，存储B。根据行列判断是否可以相加。可以的话，调用addMa函数进行相加，之后调用output函数输出C。

3）

int initial(int a,int b,TSPtr A)//初始化

{ 行数、列数初始化;

A->tu=0;

}

int enter(int u,int a,int b,int n,TSPtr A)//输入存储

{ A的data[u]的行i设为a；

A的data[u]的列j设为b；

}

int addMa(TSPtr A,TSPtr B,TSPtr C)//加法

{ C的行列=A的行列；

C中非零元素个数=A中非零元素+B中非零元素

int ca=1,cb=1,cc=1,count;

for(count<=C中非零个数)

{ if(B中非零先用完)

{ A传入C；

ca++;

cc++;

}

else if(B中非零先用完)

{ B传入C；

cb++;

cc++;

}

else if(AB中非零都用完)

break;

else

{ if(取A的元素 行小于B)

{ A传入C；

ca++;

cc++;

}

else if(B元素行小于A)

{ B传入C；

cb++;

cc++;

}

else //行数相同

{ if(A的列数小于B)

{ A传入C；

ca++;

cc++;

}

else if(B的列数小于A)

{ B传入C；

cb++;

cc++;

}

Else//行列相同

{C行列=A行列；

C元素=A+B；

cc++;

ca++;

cb++;

}

}

}

if(C元素为0)//AB都为零或相反

continue;

}

C非零元素=cc；

}

void output(TSPtr Q)//输出

{ int row,line,count,flag;

for(行)

{ for(列)

{ flag=0;

for(count<=Q.tu)

if(行列都等于ij)

{ 输出数据；

flag=1;

}

if(flag==0)

{ printf("0 ");

}

}

printf("\n");

}

}

main()

{

读取Qa的行列；

initial(a1,b1,Qa);

//读取Qa矩阵

for(i=1;i<=a1\*b1;)

{

for(r=1;r<=a1;r++)

{for(l=1;l<=b1;l++)

{ scanf("%d",&e);

if(e!=0)

{

Qa->tu++;

enter(Qa->tu,r,l,e,Qa);

}

i++;

}

}

}

读取Qb行列；

initial(a2,b2,Qb);

读取Qb矩阵；

for(i=1;i<=a2\*b2;)

{

for(r=1;r<=a2;r++)

{for(l=1;l<=b2;l++)

{ scanf("%d",&e);

if(e!=0)

{

Qb->tu++;

enter(Qb->tu,r,l,e,Qb);

}

i++;

}

}

}

if(Qa,Qb行列相等)

{ printf("可以相加\n");

initial(a1,b1,Qc);

addMa(Qa,Qb,Qc);

}

else

{ printf("ERROR!\n");

}

output(Qc);

}

* **源程序**

（矩阵相加.c）

#include<stdio.h>

#include<stdlib.h>

#define OK 1

#define MAXSIZE 12500

//三元组存储稀疏矩阵

typedef struct{

int i,j;//该非零元的行下标和列下标

int e;//数值

}Triple,\*TriplePtr;

typedef struct{

Triple data[MAXSIZE+1];//非零元三元组表，data[0]未用

int mu,nu,tu;//矩阵的行数、列数和非零元个数

}TSMatrix,\*TSPtr;

int initial(int a,int b,TSPtr A)//初始化

{ A->mu=a;//行数初始化

A->nu=b;//列数

A->tu=0;

return OK;

}

int enter(int u,int a,int b,int n,TSPtr A)//输入存储

{ A->data[u].i=a;

A->data[u].j=b;

A->data[u].e=n;

return OK;

}

int addMa(TSPtr A,TSPtr B,TSPtr C)//加法

{ C->mu=A->mu;//行

C->nu=A->nu;//列

C->tu=A->tu+B->tu;//非零元素个数

int ca=1,cb=1,cc=1,count;

for(count=1;count<=C->tu;count++)

{ if(ca<=A->tu&&cb>B->tu)//B中非零元素先完

{ C->data[cc]=A->data[ca];

ca++;

cc++;

}

else if(ca>A->tu&&cb<=B->tu)//A中非零元素先完

{ C->data[cc]=B->data[cb];

cb++;

cc++;

}

else if(ca>A->tu&&cb>B->tu)//AB中非零元素都完

{ break;

}

else //AB中非零元素都没有完

{ if(A->data[ca].i<B->data[cb].i)//A的行小于B的行

{ C->data[cc]=A->data[ca];//A传入C

ca++;

cc++;

}

else if(A->data[ca].i>B->data[cb].i)//B的行小于A的行

{ C->data[cc]=B->data[cb];//B传入C

cb++;

cc++;

}

else //A的行等于B的行

{ if(A->data[ca].j<B->data[cb].j)//A的列小于B的列

{ C->data[cc]=A->data[ca];//A传入C

ca++;

cc++;

}

else if(A->data[ca].j>B->data[cb].j)//B的列小于A的列

{ C->data[cc]=B->data[cb];//B传入C

cb++;

cc++;

}

else//AB行列相等，进行相加

{ C->data[cc].i=A->data[ca].i;

C->data[cc].j=A->data[ca].j;

C->data[cc].e=B->data[cb].e+A->data[ca].e;

cc++;

ca++;

cb++;

}

}

}

if(C->data[cc].e==0)//AB都为0或为相反数

continue;

}

C->tu=cc;

}

void output(TSPtr Q)

{ int row,line,count,flag;

for(row=1;row<=Q->mu;row++)

{ for(line=1;line<=Q->nu;line++)

{ flag=0;

for(count=1;count<=Q->tu;count++)

if(Q->data[count].i==row&&Q->data[count].j==line)//对应位置有数

{ printf("%d ",Q->data[count].e);//打印数据

flag=1;

}

if(flag==0)

{ printf("0 ");//否则打印0

}

}

printf("\n");

}

}

main()

{ int i,a1,b1,a2,b2,e,r,l;

TSPtr Qa,Qb,Qc;

Qa=(TSPtr)malloc(sizeof(TSMatrix));

Qb=(TSPtr)malloc(sizeof(TSMatrix));

Qc=(TSPtr)malloc(sizeof(TSMatrix));

printf("Qa row:");

scanf("%d",&a1);

printf("Qa line:");

scanf("%d",&b1);//读取Qa行列

initial(a1,b1,Qa);//初始化Qa

printf("Please input Qa:\n");

for(i=1;i<=a1\*b1;)//存储Qa矩阵

{

for(r=1;r<=a1;r++)

{for(l=1;l<=b1;l++)

{ scanf("%d",&e);

if(e!=0)

{

Qa->tu++;

enter(Qa->tu,r,l,e,Qa);

}

i++;

}

}

}

// output(Qa);

printf("\n\nQb row:");

scanf("%d",&a2);

printf("Qb line:");

scanf("%d",&b2);//读取Qb行列

initial(a2,b2,Qb);//初始化Qb

printf("Please input Qb:\n");

for(i=1;i<=a2\*b2;)//存储Qb矩阵

{

for(r=1;r<=a2;r++)

{for(l=1;l<=b2;l++)

{ scanf("%d",&e);

if(e!=0)

{

Qb->tu++;

enter(Qb->tu,r,l,e,Qb);

}

i++;

}

}

}

if(Qa->mu==Qb->mu&&Qa->nu==Qb->nu)//判断Qa，Qb是否行列相同

{ printf("可以相加\n");

initial(a1,b1,Qc);

addMa(Qa,Qb,Qc);//相加运算

}

else

{ printf("ERROR!\n");

}

output(Qc);

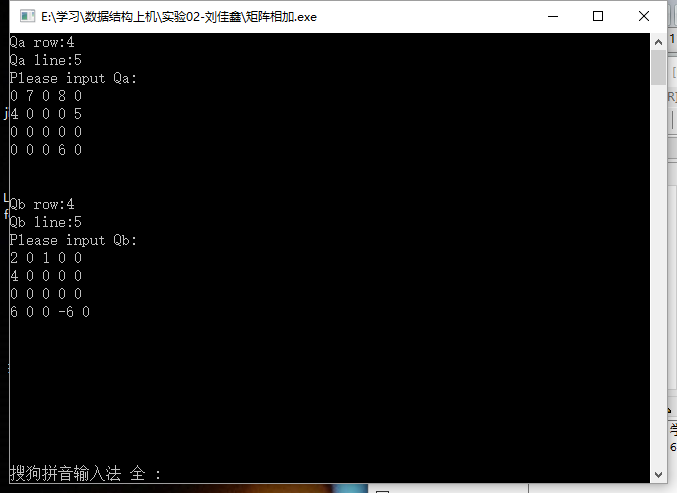
system("pause");

return 0;

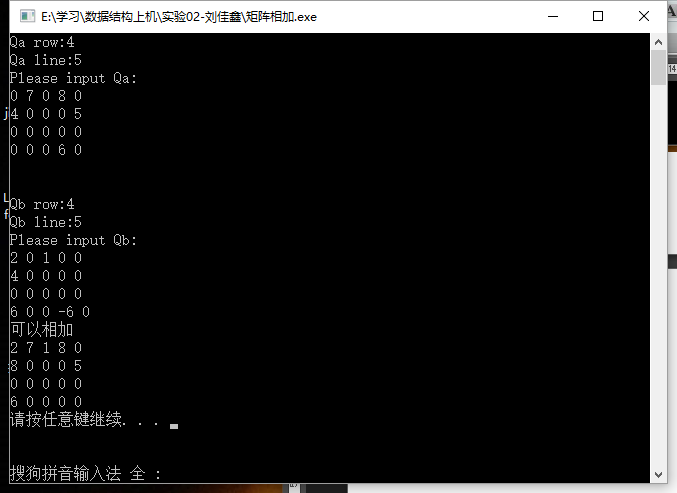
}

* **测试结果**

行列相等时：输入

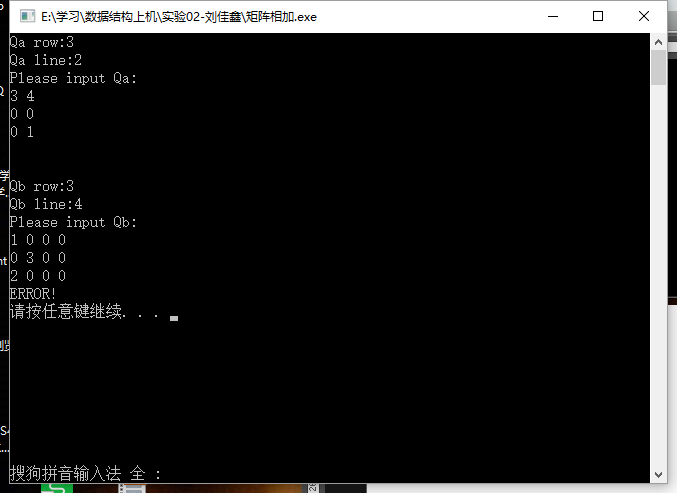


输出



行列不等时

输入输出：输出为ERROR！



* **用户使用说明：**

运行环境：Dev C++

输入要求：按提示输入行数和列数，然后按照格式或者按照矩阵元素个数（行\*列）输入矩阵数据。

* **心得体会**：

开始时对三元组表理解不深，在加法时不太会遍历矩阵寻找需要相加的元素，之后从网上查了下，看了别人的算法，对比了一下，选择了这种算法。然后else和else if用乱了，调试时改过来的。

感觉自己在写算法时，思路比较混乱，也是知识点不熟的原因，对三元组的存储方式不熟，所以开始不会遍历寻找合适元素，以及不会遍历输出。所以以后要先对于存储结构大概画个图。