中国科学院大学
University of Chinese Academy of Sciences

2016-2017 春季学期 数据结构课程 实验报告

稀疏矩阵运算器

1511	李　勰	2015K80099*****
1511	徐易难	2015K80099*****
1511	张远航	2015K80099*****

2016~2017 学年第二学期

2017 年 7 月 5 日

目录

人员分工

李龖：编写稀疏矩阵运算库中的十字链表和 CSR 压缩存储及相关算术操作，实现命令行界面；撰写报告；

张远航：编写稀疏矩阵运算库中的向量及三元组存储，辅助李龖实现代数余子式；用 LaTeX 完成实验报告和 PPT 的撰写；课堂汇报；

徐易难：完成多关键字排序的相关代码并实现 Python 可视化，观察关键字和记录个数增加时的现象并分析其成因；撰写报告。

实验一　稀疏矩阵运算器

1　需求分析

1. 实现以“带行逻辑链接的”三元组表示的稀疏矩阵运算库，要求具有加法、减法、矩阵乘法运算。
2. 稀疏矩阵的存储要求能够节省存储空间，提高运算效率。
3. 形成以十字链表为数据结构的稀疏矩阵库。
4. 实现矩阵的求逆运算。
5. 实现一维向量，二维向量及相关运算操作。
6. 实现现代稀疏矩阵的存储结构之一 CSR（Compressed Sparse Row），及相应的加法、减法和乘法运算。
7. 测试数据：练习册上的测试数据，以及线性代数课本中的一些矩阵。

2　概要设计

2.1　稀疏矩阵的抽象数据类型定义

此处给出实习题目中要求的带行链接三元组存储方式的抽象定义。

```
ADT SparseMatrix_RowLinkedTriple{
    数据对象：$D = \{a_{ij} \mid a_{ij} \in \text{ElemSet}, i = 1, 2, \ldots, n, n \geq 0, j = 1, 2, \ldots, m\}$，其中$n$和$m$
        分别为矩阵的行数和列数
    数据关系：$R_1 = \{\langle a_{i-1,j}, a_{ij}\rangle \mid a_{i-1,j}, a_{ij} \in D, i = 2, \ldots, n, j = 1, 2, \ldots, m\}$，
        $R_2 = \{\langle a_{i,j-1}, a_{ij}\rangle \mid a_{i,j-1}, a_{ij} \in D, i = 2, \ldots, n, j = 1, 2, \ldots, m\}$
    基本操作：
        CreateMatrix(M, r, c, ne);
        操作结果：创建一个稀疏矩阵M，行数为r，列数为c，非零元有ne个。
        AssignMatrix(&M)
        初始条件：稀疏矩阵已创建
        操作结果：将稀疏矩阵M的值赋给当前矩阵
        PrintMatrix(&M)
        操作结果：在命令行界面输出稀疏矩阵M
```

　　==（运算符）
　　操作结果：如果两个稀疏矩阵相等，返回为真，否则为假
　　+（运算符）
　　操作结果：返回两个矩阵的和
　　*（运算符）
　　操作结果：返回两个矩阵的乘积
　　TransposeMatrix(&M)
　　操作结果：返回矩阵M的转置
　　Determinant(&M)
　　初始条件：矩阵为方阵（即r=c）
　　操作结果：返回矩阵M的行列式
　　Inverse(&M)
　　初始条件：$\det M \neq 0$
　　操作结果：返回矩阵M的逆（如果存在）
　　DestroyMatrix()
　　初始条件：稀疏矩阵存在
　　操作结果：析构稀疏矩阵
}ADT SparseMatrix_RowLinkedTriple

3　详细设计

3.1　一维向量 Array1D

```
class Array1d
{
   public:
      Array1d(unsigned int length, bool rowVector);
      Array1d();
      void resizeArray(unsigned arySize, T value);
      void resizeArray(unsigned arySize);
      void setZero();
      void elemAssign(unsigned int pos, T value);
      void transpose();
      void arrayPrint() const;
      unsigned int getSize() const;
      T dotProduct(const Array1d<T> &ary) const;

      Array1d operator+(const Array1d<T> &A);
      virtual ~Array1d();
      std::vector<T> vec;
   protected:
      bool rowVec;
      unsigned int arraySize;

   private:
};
```

3.2　二维向量 Array2D

```
class Array2d
{
   public:
      Array2d();
      Array2d(unsigned int wid, unsigned int hgt);
      void printMatrix() const;
      unsigned int getWidth() const;
      unsigned int getHeight() const;
      unsigned int getNonZeroNum() const;
      Array2d getNegMatrix() const;
      Array2d transpose() const;

      void insertElem(unsigned int row, unsigned int col, T val);

      void operator=(const Array2d &M);
      bool operator==(const Array2d &M) const;
      Array2d operator+(const Array2d &M);
      Array2d operator-(const Array2d &M);
      Array2d operator*(const Array2d &M);


      virtual ~Array2d();
      std::vector<std::vector<T>> data;
   protected:
      unsigned int matrixWidth; unsigned int matrixHeight;
   private:
};
```

3.3　十字链表稀疏矩阵 CLMatrix

矩阵定义如下：

```
template <typename T>
class CLMatrix
{
   public:
      CLMatrix();
      CLMatrix(unsigned int wid, unsigned int hgt, unsigned int nonZero);
      virtual ~CLMatrix();

      void insertNode(CLNode<T> ins);
      void insertNode(unsigned int row, unsigned int col, T val);
      void printMatrix() const;
      void operator=(const CLMatrix &M);
      bool operator==(const CLMatrix &M) const;
      CLMatrix getNegMat() const;
      CLMatrix operator+(const CLMatrix &M);
      CLMatrix operator*(const CLMatrix &M);
      CLMatrix operator-(const CLMatrix &M);

      unsigned int getWidth() const;
      unsigned int getHeight() const;
      unsigned int getNonZero() const;
      T getDeterminant();
```

```
    T naive_getCofactor(unsigned int row, unsigned int col);
    CLMatrix getInverseMatrix();

    std::vector<std::shared_ptr<CLNode<T>>> rowHead;
    std::vector<std::shared_ptr<CLNode<T>>> colHead;
    std::vector<int> used;
  protected:
    unsigned int width; unsigned int height;
    unsigned int nonZeroNum;
  private:
    T deterCal(unsigned int colNow);
};
```

链表节点定义如下：

```
template <typename T>
class CLNode
{
  public:
    CLNode();
    CLNode(unsigned int row, unsigned int col, T val);
    virtual ~CLNode();
    void modifyNode(unsigned int row, unsigned int col, T val);
    void createDownNode(unsigned int row, unsigned int col, T val);
    void createRightNode(unsigned int row, unsigned int col, T val);
    void deleteDownNode();
    void deleteRightNode();

    void printNode() const;
    void printRightList() const;
    void printDownList() const;
    void printRightListVal(unsigned int hgt) const;

    unsigned int getRowNum() const;
    unsigned int getColNum() const;
    T getVal() const;

    void operator=(const CLNode &N);
    std::shared_ptr<CLNode> down;
    std::shared_ptr<CLNode> right;
  protected:
    unsigned int rowNum;
    unsigned int colNum;
    T value;
  private:
};
```

3.4 三元组矩阵 TripletMatrix（RowLinkTriMat）

三元组定义如下：

```
template <typename T>
class Triplet
{
  public:
    Triplet();
    /* ATTENTION:
     * row number of first row is 1
     * 0 means the end of the list.
     */
    virtual ~Triplet();
    // Methods that do no change to instance: add const qualifier.
    unsigned int getRowNum() const;
    unsigned int getColNum() const;
    T getValue() const;
    void displayTriplet() const;

    void modifyTriplet(unsigned int newRow, unsigned int newCol, T newVal);
    void operator=(const Triplet &t);
    bool operator==(const Triplet &t) const;
    Triplet operator+(const Triplet &t);
    Triplet operator-(const Triplet &t);
  protected:
    unsigned int rowNum;
    unsigned int colNum;
    T value;
  private:
};
```

纯三元组存储的矩阵定义如下:

```
template <typename T>
class TripletMatrix
{
public:
TripletMatrix();
virtual ~TripletMatrix();
// Instance untouched
void printMatrix() const;
void displayTable() const;
unsigned int getMatrixWidth() const;
unsigned int getMatrixHeight() const;
unsigned int getMatrixNonZeroNum() const;
TripletMatrix getNegMatrix() const;
TripletMatrix transposeMatrix() const;
// Instance Modifying
void resizeMatrix(unsigned int width, unsigned int height, unsigned int nonZero);
void insertTripletToMatrix(Triplet<T> insertTriplet);
void nonZeroUpdate();
// Operators Overload
void operator=(const TripletMatrix &M);
bool operator==(const TripletMatrix &M) const;
TripletMatrix operator+(const TripletMatrix &M);
TripletMatrix operator-(const TripletMatrix &M);
TripletMatrix operator*(const TripletMatrix &M);
std::vector<Triplet<T>> data;
protected:
unsigned int matrixWidth; unsigned int matrixHeight;
unsigned int nonZeroNum;

private:
};
```

带行链接的三元组矩阵定义如下：

```
template <typename T>
```

```cpp
class RowLinkTriMat : public TripletMatrix<T>
{
public:
RowLinkTriMat();
virtual ~RowLinkTriMat();
void resizeMatrix(unsigned int width, unsigned int height, unsigned int nonZero);
void insertTripletToMatrix(Triplet<T> insertTriplet);
void displayTable() const;
void operator=(const RowLinkTriMat &M);
RowLinkTriMat operator+(const RowLinkTriMat &M);
RowLinkTriMat operator*(const RowLinkTriMat &M);
protected:
std::vector<unsigned int> rowPtr;
private:
};
```

3.5 CSR 压缩存储 CSRStore

矩阵定义如下：

```cpp
template <typename T>
class CSRMatrix
{
    public:
        CSRMatrix(unsigned int width, unsigned int height);
        virtual ~CSRMatrix();
        void printMatrix() const;
        void displayTable() const;

        unsigned int getMatrixWidth() const;
        unsigned int getMatrixHeight() const;
        unsigned int getMatrixNonZeroNum() const;
        CSRMatrix getNegMat() const;
        //instance modifying
        void clearCSRMatrix(unsigned int matWid, unsigned int matHgt);
        void insertTupleToMatrix(unsigned int rowNum, CSRTuple<T> ins);
        void insertElemToMat(unsigned int rowNum, unsigned int colNum, T value);
        void addInsert(unsigned int rowNum, CSRTuple<T> ins);
        //operator overload
        void operator=(const CSRMatrix<T> &M);
        bool operator==(const CSRMatrix<T> &M) const;
        CSRMatrix operator+(const CSRMatrix<T> &M);
        CSRMatrix operator-(const CSRMatrix<T> &M);
        CSRMatrix operator*(const CSRMatrix<T> &M);
    protected:
        unsigned int matrixWidth;
        unsigned int matrixHeight;
        std::vector<int>rowPtr;
        std::vector<CSRTuple<T>> data;
    private:
};
```

三元组定义如下：

```cpp
template <typename T>
class CSRTuple
{
   public:
       CSRTuple();
       virtual ~CSRTuple();
       void modifyTuple(unsigned int col, T val);
       void destroyTuple();
       void printTuple() const;
       unsigned int getColNum() const;
       T getVal() const;
       void operator=(const CSRTuple &t);
       bool operator!=(const CSRTuple &t) const;
       bool operator==(const CSRTuple &t) const;
       CSRTuple operator+(const CSRTuple &t);
       CSRTuple operator-(const CSRTuple &t);
   protected:
       unsigned int colNum;
       T value;
   private:
};
```

3.6 求逆部分核心代码段

```cpp
// 求矩阵的行列式，用Laplace展开可以很好地利用稀疏矩阵非零元较多的特性
template <typename T>
T CLMatrix<T>::deterCal(unsigned int colNow) {
    // 已经缩小为2x2子式，可以直接求行列式
    if(colNow == this->getHeight()-1) {
        unsigned int detIndex[2] = {0, 0};
        T subMat[2][2] = {0};
        for(unsigned int i = 1; i < this->used.size(); i++) {
            if(this->used[i]) {
                if(!detIndex[0]) detIndex[0] = i;
                else detIndex[1] = i;
            }
        }
        std::shared_ptr<CLNode<T>> temp;
        temp = this->colHead[colNow]->down;
        while(temp) {
            // 跳过原矩阵不在子式中的行
            if(this->used[temp->getRowNum()] == 1) {
                if(detIndex[0] == temp->getRowNum())
                    subMat[0][0] = temp->getVal();
                else
                    subMat[0][1] = temp->getVal();
            }
            temp = temp->down;
        }
        temp = this->colHead[colNow+1]->down;
        while(temp) {
            if(this->used[temp->getRowNum()] == 1) {
                if(detIndex[0] == temp->getRowNum())
                    subMat[1][0] = temp->getVal();
                else subMat[1][1] = temp->getVal();
            }
            temp = temp->down;
        }
        return (subMat[0][0]*subMat[1][1] - subMat[1][0]*subMat[0][1]);
    }
    else {
        T det = 0;
        std::shared_ptr<CLNode<T>> temp = this->colHead[colNow]->down;
        while(temp) {
            if (this->used[temp->getRowNum()]) {
                int parity = ((temp->getRowNum() + colNow) & 0x1)? -1: 1;
                this->used[temp->getRowNum()] = 0;
                // 按列递归
                det += parity * temp->getVal() * deterCal(colNow+1);
                this->used[temp->getRowNum()] = 1;
            }
            temp = temp->down;
        }
        return det;
    }
}

// 求代数余子式的朴素实现，类似上面
template <typename T>
T CLMatrix<T>::naive_getCofactor(unsigned int row, unsigned int col) {
    CLMatrix<T> Temp(this->getWidth()-1, this->getHeight()-1, 0);
    for(unsigned int i = 1; i < row; i++) {
        std::shared_ptr<CLNode<T>> temp = rowHead[i]->right;
        while(temp) {
            if(temp->getColNum() < col)
                Temp.insertNode(temp->getRowNum(),
                                temp->getColNum(),
                                temp->getVal());
            else {
                if(temp->getColNum() == col){;}
                else Temp.insertNode(temp->getRowNum(),
                                     temp->getColNum()-1,
                                     temp->getVal());
            }
            temp = temp->right;
        }
    }
    for(unsigned int i = row+1; i <= this->getWidth(); i++) {
        std::shared_ptr<CLNode<T>> temp = rowHead[i]->right;
        while(temp) {
            if(temp->getColNum() < col)
                Temp.insertNode(temp->getRowNum()-1,
                                temp->getColNum(),
                                temp->getVal());
            else {
                if(temp->getColNum() == col){;}
                else{Temp.insertNode(temp->getRowNum()-1,
                                     temp->getColNum()-1,
                                     temp->getVal());}
            }
            temp = temp->right;
        }
    }
    int parity = ((row + col) & 0x1)? -1: 1;
    return parity * Temp.getDeterminant();
}
```

```cpp

// 矩阵求逆
template <class T>
CLMatrix<T> CLMatrix<T>::getInverseMatrix() {
    CLMatrix<T> Temp(this->getHeight(),
                     this->getWidth(), 0);
    T det = this->getDeterminant();
    // 判断行列式是否为0
    if(!det)
    {
        std::cout << "CLMatrix not invertible." << '\n';
    }
    // 先求余子式，再求伴随矩阵
    for(unsigned int i = 1; i < this->rowHead.size(); i++) {
        std::shared_ptr<CLNode<T>> temp = this->rowHead[i]->right;
        while(temp) {
            double coef = (naive_getCofactor(i, temp->getColNum())/det);
            Temp.insertNode(temp->getColNum(), i, coef);
            temp = temp->right;
        }
    }
    return Temp;
}
```

4　调试分析

实现难点：

稀疏矩阵运算库的实现难点主要是每一种稀疏矩阵的存储结构中的插入操作，只要能够将元素的插入问题解决，其他的运算都不会有太大的问题。

程序优点：

1. 实现了多种存储结构。
2. 利用 C++ 的 `shared_ptr` 智能指针类，不会出现内存泄漏。
3. 程序有专门编写的 CLI (Command Line Interface)，便于演示操作。
4. 该库的可扩展性强，能够利用已有的基本运算来进行复杂的运算，能够继续添加更多的数据结构。

程序的局限性：

1. 运算操作不够丰富。
2. 没有实现 GUI（目前只有命令行演示）。

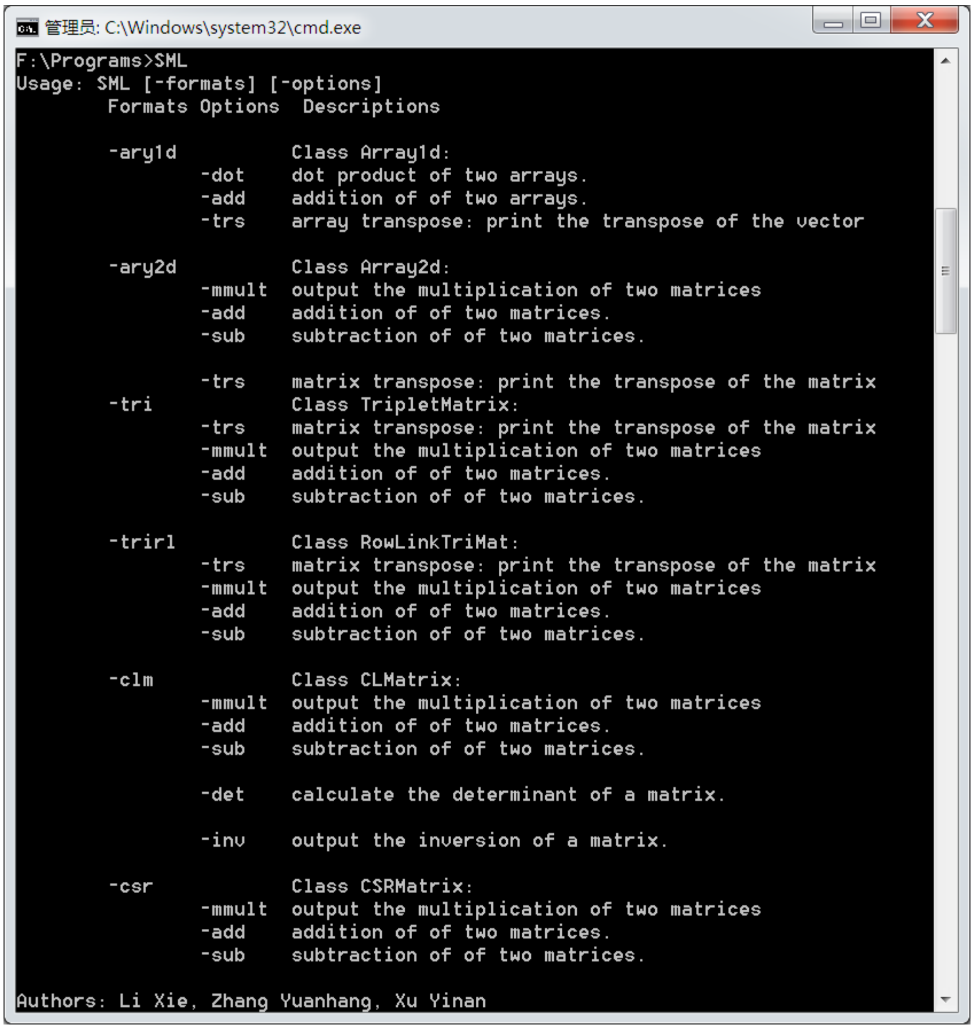
5　用户手册

1. 本程序运行的环境为 Windows 10，可执行文件为：`SML.exe`。
2. 进入演示程序后，即显示命令行方式的用户界面。

3. 按照提示在命令行输入参数，可以进行相应的运算程序，其中有提示输入内容。
4. 按要求输入全部数据后，即返回依次出列的结果。
5. 程序执行完毕后显示作者信息。

6 测试结果

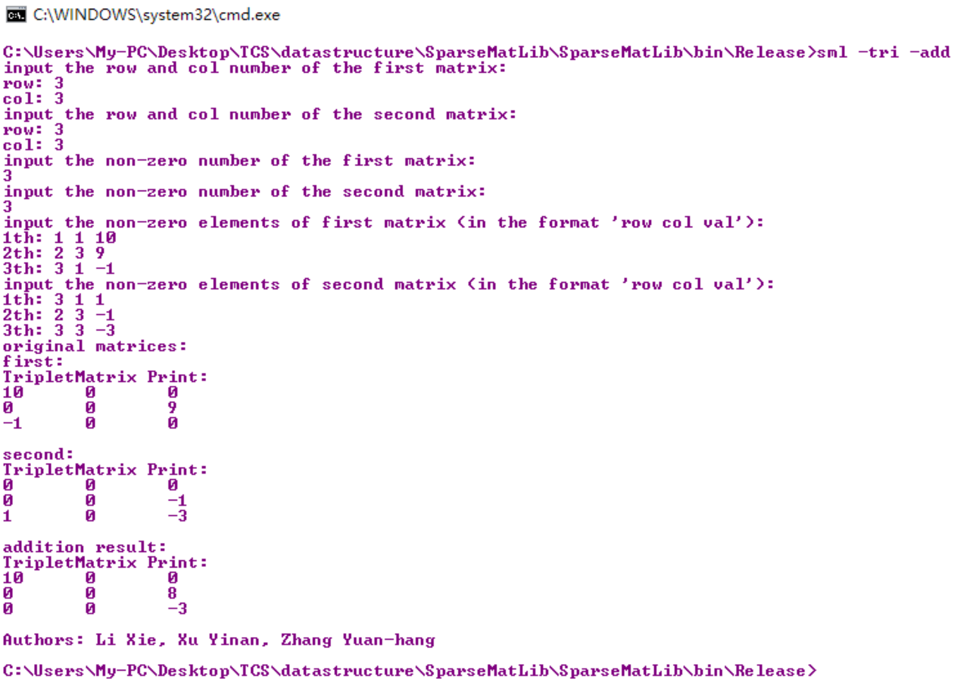
命令行提示界面如下：



```
F:\Programs>SML
Usage: SML [-formats] [-options]
        Formats Options  Descriptions

        -ary1d          Class Array1d:
                -dot    dot product of two arrays.
                -add    addition of of two arrays.
                -trs    array transpose: print the transpose of the vector

        -ary2d          Class Array2d:
                -mmult  output the multiplication of two matrices
                -add    addition of of two matrices.
                -sub    subtraction of of two matrices.

                -trs    matrix transpose: print the transpose of the matrix
        -tri            Class TripletMatrix:
                -trs    matrix transpose: print the transpose of the matrix
                -mmult  output the multiplication of two matrices
                -add    addition of of two matrices.
                -sub    subtraction of of two matrices.

        -trirl          Class RowLinkTriMat:
                -trs    matrix transpose: print the transpose of the matrix
                -mmult  output the multiplication of two matrices
                -add    addition of of two matrices.
                -sub    subtraction of of two matrices.

        -clm            Class CLMatrix:
                -mmult  output the multiplication of two matrices
                -add    addition of of two matrices.
                -sub    subtraction of of two matrices.

                -det    calculate the determinant of a matrix.

                -inv    output the inversion of a matrix.

        -csr            Class CSRMatrix:
                -mmult  output the multiplication of two matrices
                -add    addition of of two matrices.
                -sub    subtraction of of two matrices.

Authors: Li Xie, Zhang Yuanhang, Xu Yinan
```

部分测试数据和输出结果如下：

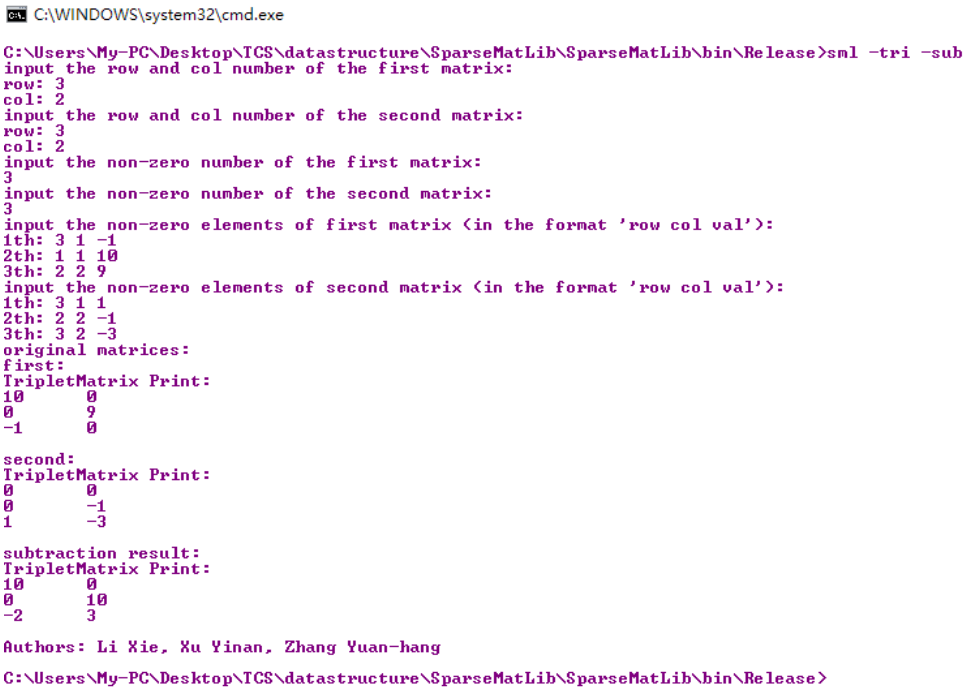
1. 按照提示依次键入：`sml -tri -add`，使用三元组矩阵进行加法。输入两个矩阵的行数和列数，两个矩阵的非零元素个数和非零元素。输出结果如图。



C:\WINDOWS\system32\cmd.exe

```
C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>sml -tri -add
input the row and col number of the first matrix:
row: 3
col: 3
input the row and col number of the second matrix:
row: 3
col: 3
input the non-zero number of the first matrix:
3
input the non-zero number of the second matrix:
3
input the non-zero elements of first matrix (in the format 'row col val'):
1th: 1 1 10
2th: 2 3 9
3th: 3 1 -1
input the non-zero elements of second matrix (in the format 'row col val'):
1th: 3 1 1
2th: 2 3 -1
3th: 3 3 -3
original matrices:
first:
TripletMatrix Print:
10      0       0
0       0       9
-1      0       0

second:
TripletMatrix Print:
0       0       0
0       0       -1
1       0       -3

addition result:
TripletMatrix Print:
10      0       0
0       0       8
0       0       -3

Authors: Li Xie, Xu Yinan, Zhang Yuan-hang

C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>
```

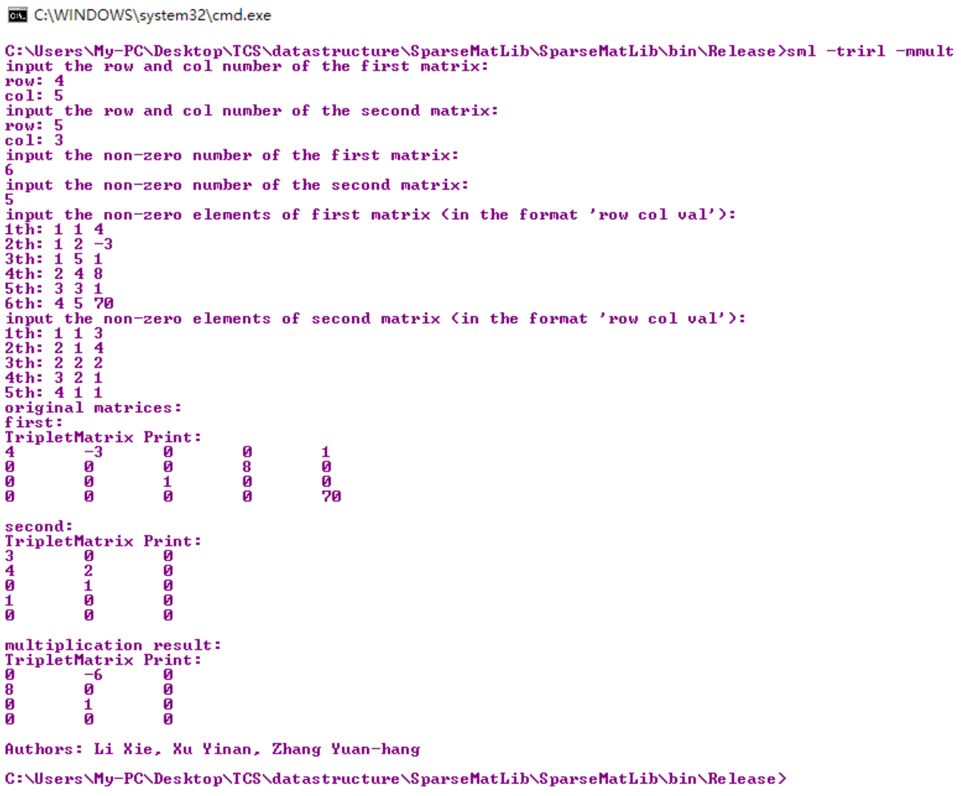
2. 按照提示依次键入：`sml -tri -sub`，使用三元组矩阵进行减法。输入两个矩阵的行数和列数，两个矩阵的非零元素个数和非零元素。输出结果如图。



C:\WINDOWS\system32\cmd.exe

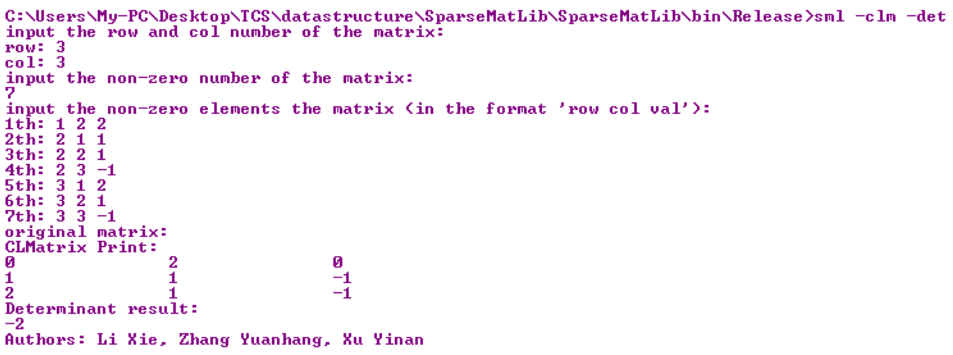
```
C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>sml -tri -sub
input the row and col number of the first matrix:
row: 3
col: 2
input the row and col number of the second matrix:
row: 3
col: 2
input the non-zero number of the first matrix:
3
input the non-zero number of the second matrix:
3
input the non-zero elements of first matrix (in the format 'row col val'):
1th: 3 1 -1
2th: 1 1 10
3th: 2 2 9
input the non-zero elements of second matrix (in the format 'row col val'):
1th: 3 1 1
2th: 2 2 -1
3th: 3 2 -3
original matrices:
first:
TripletMatrix Print:
10      0
0       9
-1      0

second:
TripletMatrix Print:
0       0
0       -1
1       -3

subtraction result:
TripletMatrix Print:
10      0
0       10
-2      3

Authors: Li Xie, Xu Yinan, Zhang Yuan-hang

C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>
```

3. 按照提示依次键入：`sml -trirl -sub` 使用带行逻辑链接的三元组矩阵进行减法，输入两个矩阵的行数和列数，非零元素个数和非零元素。运行结果如图：



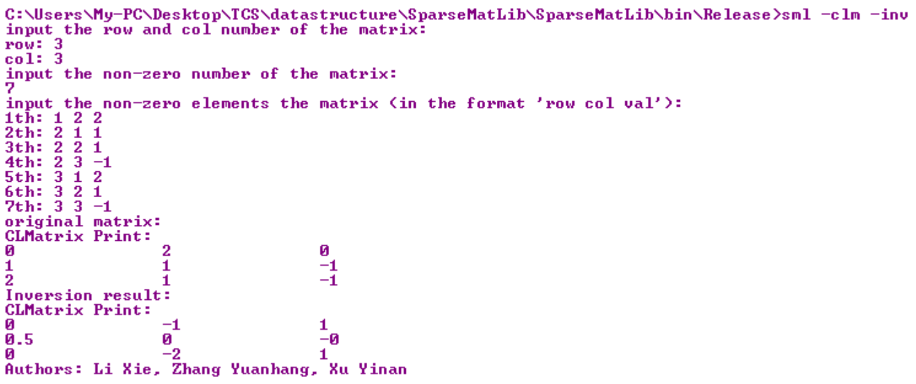
```
C:\WINDOWS\system32\cmd.exe

C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>sml -trirl -mmult
input the row and col number of the first matrix:
row: 4
col: 5
input the row and col number of the second matrix:
row: 5
col: 3
input the non-zero number of the first matrix:
6
input the non-zero number of the second matrix:
5
input the non-zero elements of first matrix (in the format 'row col val'):
1th: 1 1 4
2th: 1 2 -3
3th: 1 5 1
4th: 2 4 8
5th: 3 3 1
6th: 4 5 70
input the non-zero elements of second matrix (in the format 'row col val'):
1th: 1 1 3
2th: 2 1 4
3th: 2 2 2
4th: 3 2 1
5th: 4 1 1
original matrices:
first:
TripletMatrix Print:
4       -3      0       0       1
0       0       0       8       0
0       0       1       0       0
0       0       0       0       70

second:
TripletMatrix Print:
3       0       0
4       2       0
0       1       0
1       0       0
0       0       0

multiplication result:
TripletMatrix Print:
0       -6      0
8       0       0
0       1       0
0       0       0

Authors: Li Xie, Xu Yinan, Zhang Yuan-hang

C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>
```

4. 按照提示依次键入：`sml -clm -det` 对十字链表存储的矩阵求行列式，输入两个矩阵的行数和列数，非零元素个数和非零元素。运行结果如图：



```
C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>sml -clm -det
input the row and col number of the matrix:
row: 3
col: 3
input the non-zero number of the matrix:
7
input the non-zero elements the matrix (in the format 'row col val'):
1th: 1 2 2
2th: 2 1 1
3th: 2 2 1
4th: 2 3 -1
5th: 3 1 2
6th: 3 2 1
7th: 3 3 -1
original matrix:
CLMatrix Print:
0               2               0
1               1               -1
2               1               -1
Determinant result:
-2
Authors: Li Xie, Zhang Yuanhang, Xu Yinan
```

5. 按照提示依次键入：`sml -clm -inv` 对十字链表存储的矩阵求逆，输入两个矩阵的行数和列数，非零元素个数和非零元素。运行结果如图：



```
C:\Users\My-PC\Desktop\TCS\datastructure\SparseMatLib\SparseMatLib\bin\Release>sml -clm -inv
input the row and col number of the matrix:
row: 3
col: 3
input the non-zero number of the matrix:
7
input the non-zero elements the matrix (in the format 'row col val'):
1th: 1 2 2
2th: 2 1 1
3th: 2 2 1
4th: 2 3 -1
5th: 3 1 2
6th: 3 2 1
7th: 3 3 -1
original matrix:
CLMatrix Print:
0               2               0
1               1               -1
2               1               -1
Inversion result:
CLMatrix Print:
0               -1              1
0.5             0               -0
0               -2              1
Authors: Li Xie, Zhang Yuanhang, Xu Yinan
```

附录

文件结构如下：

```
|---include
|  |---MatLib
|      |---ArrayStore
|      |      Array1d.h
|      |      Array2d.h
|      |---CrossListStore
|      |      CLMatrix.h
|      |      CLNode.h
|      |---CSRStore
|      |      CSRMatrix.h
|      |      CSRTuple.h
|      |---TripletStore
|              RowLinkTriMat.h
|              Triplet.h
|              TripletMatrix.h
|---src
    |---MatLib
        |---ArrayStore
        |      Array1d.cpp
        |      Array2d.cpp
        |---CrossListStore
        |      CLMatrix.cpp
        |      CLNode.cpp
        |---CSRStore
        |      CSRMatrix.cpp
        |      CSRTuple.cpp
        |---TripletStore
                RowLinkTriMat.cpp
                Triplet.cpp
                TripletMatrix.cpp
   |---main.cpp
```