山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

学号：202200130048	姓名： 陈静雯	班级： 6
实验题目：数组和矩阵		
实验学时：2	实验日期： 10.26	

实验目的：
掌握稀疏矩阵结构的描述及操作的实现。

软件开发工具：
Vscode

1. 实验内容
创建稀疏矩阵类(参照课本 MatrixTerm 三元组定义)，采用行主顺序把稀疏矩阵非 0 元素映射到一维数组中，实现操作：两个稀疏矩阵相加、两个稀疏矩阵相乘、稀疏矩阵的转置、输出矩阵。
不得使用相关 STL。

2. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）
重置：按行优先遍历矩阵，保存到稀疏矩阵中
乘法：矩阵 A*B，先将 B 转置，再将 A 的每一行与 B 相乘，标记 A 的行首元素，A 的一行与 B 的一行乘完之后，A 回到行首，与 B 的下一行继续相乘。行与行相乘的时候，找到对应元素相乘后相加即可。
加法：根据 t 中的非零元素，找到被加矩阵的对应位置，两个元素相加即可
输出：根据 cols 和 rows 二重循环，若该位置有元素则输出该元素，若无则输出 0
转置：一个 colsize 数组记录每一列有多少元素，再根据该数组得到 rownext 数组，标记转置后的矩阵每一行开头的元素放入稀疏矩阵的位置，每行放入一个元素，rownext++为下一个元素该放的位置，最后得到结果矩阵。

3. 测试结果（测试输入，测试输出）
输入及输出

```
40
1
10 20
-1 0 1 0 0 0 0 0 -1 0 0 0 -1 0 -1 0 0 -1 1 -1
0 0 2 -1 0 0 0 0 0 -1 0 0 0 0 0 0 0 1 -2 0
1 0 0 0 0 0 0 0 0 0 -1 -2 -1 0 -1 0 0 0 0 0
0 0 0 1 0 -1 -1 -1 0 0 1 0 0 0 0 0 0 0 -1 0
0 0 0 1 0 0 0 0 0 1 -1 1 0 0 0 0 -1 0 0 0
1 0 -1 1 2 0 0 0 1 0 0 0 0 -1 0 1 -1 1 -1 0
-1 0 0 0 -1 0 0 0 0 0 -1 0 0 -1 2 0 0 -1 0 0
-1 -1 -1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 -3 0 0 0 0 -1 -2 1 0 2 0 -1 -1 0
```

```
-1 1 0 1 -1 0 0 0 -1 0 -1 0 0 0 0 1 0 0 -1 1
2
10 20
7
2 16 9
3 7 3
3 17 4
6 3 4
7 12 10
8 13 6
10 8 3
-1
2
10 20
8
1 20 1
4 20 5
6 5 4
6 10 10
7 4 8
7 6 10
8 12 9
9 17 5
-1
2
10 20
9
1 8 4
3 8 6
3 17 7
5 1 10
5 8 4
6 9 4
7 12 7
9 10 9
9 17 7
-1
3
10 20
7
3 3 10
5 18 4
8 5 2
8 19 5
8 20 10
9 12 3
```

```
10 11 10
4
10 20
0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 10 0 0 0 0 6 0 0 0 0 0 0 0 0 7 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 4 0 0
0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 5 10
0 0 0 0 0 0 0 0 0 9 0 3 0 0 0 0 7 0 0 0
0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0
2
10 20
2
3 16 4
4 10 6
-1
2
10 20
7
1 16 8
2 9 8
3 8 9
4 2 4
4 20 7
8 10 7
10 3 4
-1
2
10 20
1
1 19 5
-1
2
10 20
10
1 9 8
2 15 5
3 2 10
4 2 5
4 3 9
4 7 10
6 6 6
6 14 6
```

```
7 2 7
9 16 9
-1
2
10 20
7
3 14 5
4 9 8
6 19 5
7 17 7
8 13 4
9 6 10
9 20 1
-1
5
2
20 10
7
6 9 2
7 8 10
7 9 9
11 1 10
12 5 6
18 4 8
20 6 4
-1
2
20 10
2
13 2 5
17 5 10
-1
1
19 19
2 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1
0 0 1 0 1 -3 0 0 -1 1 -2 0 -2 0 0 1 0 0 0
-1 -1 0 0 1 0 0 1 0 -1 0 0 1 1 0 0 0 1 0
0 0 -1 0 0 -2 -1 0 0 0 1 0 0 1 2 -1 2 0 0
0 1 0 -1 0 0 -1 0 0 -1 0 0 0 -1 0 -1 0 -1 0
0 0 0 -1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 -1
1 0 -1 1 0 0 -1 0 1 1 0 0 0 0 1 0 1 0 -1
0 0 0 1 0 0 0 -1 0 0 0 0 0 0 0 0 -1 0 -1
0 -1 1 0 0 0 0 0 0 0 -1 0 0 0 -1 1 0 0 0
0 0 -1 0 0 0 0 1 0 -1 2 0 2 -1 -1 0 -1 0 0
1 0 0 -1 0 0 0 0 0 0 0 0 0 0 -1 1 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 -1
```

```
−1 0 0 0 2 −1 2 −2 0 0 0 −1 1 0 0 0 0 0 0
0 0 1 0 0 −1 0 0 0 0 0 0 0 0 0 −1 0 0 −2
0 0 −1 0 0 1 0 0 1 −1 0 0 0 1 0 0 0 0 1
0 0 0 0 1 −1 −1 1 1 0 0 0 0 −1 0 0 0 0 0
0 −1 0 0 0 0 2 0 0 0 0 2 2 0 0 0 0 0 −1
0 0 0 −1 0 −1 0 0 0 0 0 −1 0 0 0 0 0 0 0
0 −1 −1 0 0 1 0 −1 1 0 −1 0 0 0 0 0 0 0 1
4
19 19
2 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1
0 0 1 0 1 −3 0 0 −1 1 −2 0 −2 0 0 1 0 0 0
−1 −1 0 0 1 0 0 1 0 −1 0 0 1 1 0 0 0 1 0
0 0 −1 0 0 −2 −1 0 0 0 1 0 0 1 2 −1 2 0 0
0 1 0 −1 0 0 −1 0 0 −1 0 0 0 −1 0 −1 0 −1 0
0 0 0 −1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 −1
1 0 −1 1 0 0 −1 0 1 1 0 0 0 0 1 0 1 0 −1
0 0 0 1 0 0 0 −1 0 0 0 0 0 0 0 0 −1 0 −1
0 −1 1 0 0 0 0 0 0 0 −1 0 0 0 −1 1 0 0 0
0 0 −1 0 0 0 0 1 0 −1 2 0 2 −1 −1 0 −1 0 0
1 0 0 −1 0 0 0 0 0 0 0 0 0 0 −1 1 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 −1
−1 0 0 0 2 −1 2 −2 0 0 0 −1 1 0 0 0 0 0 0
0 0 1 0 0 −1 0 0 0 0 0 0 0 0 0 −1 0 0 −2
0 0 −1 0 0 1 0 0 1 −1 0 0 0 1 0 0 0 0 1
0 0 0 0 1 −1 −1 1 1 0 0 0 0 −1 0 0 0 0 0
0 −1 0 0 0 0 2 0 0 0 0 2 2 0 0 0 0 0 −1
0 0 0 −1 0 −1 0 0 0 0 0 −1 0 0 0 0 0 0 0
0 −1 −1 0 0 1 0 −1 1 0 −1 0 0 0 0 0 0 0 1
2
19 19
6
5 5 2
5 17 5
12 3 3
13 15 5
14 3 5
15 9 7
2
19 19
8
7 9 1
10 1 6
12 2 4
14 3 9
14 8 2
16 7 3
```

```
18 1 1
18 14 4
2
19 19
9
1 5 3
1 18 10
4 15 4
6 7 9
11 19 6
12 2 1
14 7 6
14 14 2
17 9 8
2
19 19
7
4 18 7
5 9 1
7 2 6
11 9 3
12 16 3
15 9 2
16 5 5
2
19 19
3
3 12 4
17 7 5
18 16 4
5
2
19 19
1
17 17 2
3
19 19
6
11 8 5
11 14 5
12 19 6
17 5 4
17 15 6
19 19 4
2
19 19
```

```
7
1  1  4
4  12  5
6  1  9
7  8  3
9  18  8
13  12  2
16  14  2
2
19  19
2
8  11  7
12  4  8
3
19  19
7
1  16  5
3  9  6
5  15  3
14  14  10
15  9  6
15  14  3
15  19  7
2
19  19
6
1  19  2
5  8  6
6  16  6
9  6  6
10  18  9
15  7  5
5
5
2
19  19
6
6  7  1
10  7  6
13  5  5
15  16  6
17  9  10
19  15  3
2
19  19
6
```

```
3 5 4
4 9 5
5 15 1
11 3 5
17 5 6
17 7 7
2
19 19
1
14 6 7
5
2
19 19
3
3 10 8
4 18 1
15 15 8
2
19 19
4
5 8 10
6 9 10
6 16 6
14 15 4
5
4
19 19
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
2
19 19
9
2 17 3
4 18 9
12 3 8
13 11 10
13 19 7
14 12 4
15 4 9
17 8 9
19 4 5
2
19 19
1
7 17 6
```

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

myarray<matrixterm<T>>::iterator 编译不通过

myarray<matrixterm>::iterator 编译通过

原因：编译器不能判断 myarray<matrixterm<T>>是个类名，

所以改成 typename myarray<matrixterm<T>>即可通过

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
#include <iostream>
using namespace std;

template<class T>
struct matrixterm{
    int row;
    int col;
    T value;
};

template <class T>
class myarray{
public:
    myarray(int capacity=20);
    void resetsize(int newsize);
    void copy(myarray<T> &b);
    void insert(T& newelement);
    void set(int index,T& newelement);
    void clear();
    int arraysize(){
        return size;
    }
```

```
    class iterator{
    public:
        iterator(T* theposition){ position=theposition; }
        T& operator*() const { return *position; }
        T* operator->() const { return & *position; }
        iterator& operator++(){                              //前++
            ++position;
            return *this;
        }
        iterator operator++(int){                           //后++
            iterator last = *this;
            ++position;
            return last;
        }
        iterator& operator--(){                              //前--
            --position;
            return *this;
        }
        iterator operator--(int){                           //后--
            iterator last = *this;
            --position;
            return last;
        }
        bool operator!=(const iterator theiter) const{
            return position!=theiter.position ;
        }
        bool operator==(const iterator theiter) const{
            return position==theiter.position;
        }
    private:
        T* position;
    };

    iterator begin(){
        return iterator(element);
    }
    iterator end(){
        return iterator(element+size);
    }

    ~myarray();
private:
    int length;
    T* element;
    int size;
```

```
};

template <class T>
myarray<T>::myarray(int capacity){   //构造函数
    element = new T[capacity];
    length=capacity;
    size=0;
}

template<class T>
void myarray<T>::resetsize(int nowsize){
    if(nowsize > length){                    //若空间不够，重新进行动态分配
        length=nowsize;
        T* temp = new T [length];
        for(int i=0;i<size;i++){
            temp[i]=element[i];
        }
        T* p =  element;
        element = temp;
        delete [] p;
    }
    size=nowsize;
}

template<class T>
void myarray<T>::copy(myarray<T> &b){
    length=b.length;
    size=b.size;
    T* p =  element;
    element = new T [length];
    for(int i=0;i<size;i++){
        element[i]=b.element[i];
    }
    delete [] p;
}

template <class T>
void myarray<T>::insert(T& newelement){   //在数组最后插入新元素
    if(size>=length){                    //若空间不够，重新进行动态分配
        length*=2;
        T* temp = new T [length];
        for(int i=0;i<size;i++){
            temp[i]=element[i];
        }
        T* p =  element;
        element = temp;
```

```
        delete [] p;
    }
    element[size++]=newelement;     //队尾插入
}

template<class T>
void myarray<T>::set(int index,T& newelement){
    element[index]=newelement;
}

template<class T>
void myarray<T>::clear(){
    T* p =  element;
    element = new T [length];
    delete [] p;
    size=0;
}

template <class T>
myarray<T>::~myarray(){
    size=0,length=0;
    T* p =  element;
    element = NULL;
    delete [] p;
}


template<class T>
class mysparsematrix{
public:
    myarray<matrixterm<T>>terms;       //terms 是公有成员，方便使用
    mysparsematrix(){}
    void set();
    void reset();
    void copy(mysparsematrix<T> &b);
    void add(mysparsematrix<T> &b);
    void transpose();
    void multipul(mysparsematrix<T> &b);
    void output();
    ~mysparsematrix(){ terms.~myarray(); rows=0; cols=0; }
private:
    int rows;
    int cols;

};
```

```
template<class T>
void mysparsematrix<T>::set(){  //稀疏矩阵输入
    terms.clear();
    cin>>rows>>cols;
    int t;
    cin>>t;
    for(int i=0;i<t;i++){
        matrixterm<T> temp;
        cin>>temp.row>>temp.col>>temp.value;
        terms.insert(temp);
    }
}

template<class T>
void mysparsematrix<T>::reset(){   //普通矩阵输入
    terms.clear();
    cin>>rows>>cols;
    for(int i=1;i<=rows;i++){
        for(int j=1;j<=cols;j++){
            int t;
            cin>>t;
            if(t!=0){
                matrixterm<T> temp;
                temp.row=i;
                temp.col=j;
                temp.value=t;
                terms.insert(temp);
            }
        }
    }
}

template <class T>
void mysparsematrix<T>::copy(mysparsematrix<T> &b){ //复制矩阵
    rows=b.rows;
    cols=b.cols;
    terms.copy(b.terms);
}

template <class T>
void mysparsematrix<T>::transpose(){
    mysparsematrix<T>b;
    b.cols=rows;
    b.rows=cols;
    b.terms.resetsize(terms.arraysize());
    int* colsize = new int [cols+1];  //每一列有多少个元素
```

```
    int* rownext = new int [cols+1];  //每一行第一个元素在第几个位子
    for(int i=0;i<=cols;i++) colsize[i]=0;
    for(typename                                    myarray<matrixterm<T>>::iterator
i=terms.begin();i!=terms.end();i++) colsize[(*i).col]++;
    rownext[1]=0;
    for(int i=2;i<=cols;i++) rownext[i]=colsize[i-1]+rownext[i-1];
    matrixterm<T> temp;
    for(typename                                    myarray<matrixterm<T>>::iterator
i=terms.begin();i!=terms.end();i++){
        int j = rownext[(*i).col]++;    //找到对应的位子，rownext++表示下一次这
一排的元素该放的位子
        temp.col=(*i).row;
        temp.row=(*i).col;
        temp.value=(*i).value;
        b.terms.set(j,temp);
    }
    this->copy(b);
}

template<class T>
void mysparsematrix<T>::add(mysparsematrix<T> &b){
    if(rows!=b.rows || cols!=b.cols){
        this->copy(b);
        cout<<-1<<endl;
        return ;
    }
    mysparsematrix<T>c;
    c.rows = rows;
    c.cols = cols;
    c.terms.clear();
    typename myarray<matrixterm<T>>::iterator ithis = terms.begin();
    typename myarray<matrixterm<T>>::iterator ib = b.terms.begin();
    while(ithis!=terms.end() && ib!=b.terms.end()){
        int thisindex = ((*ithis).row - 1) * cols + (*ithis).col;  //行列都是从
1 开始算
        int bindex = ((*ib).row - 1) * cols + (*ib).col;
        if(thisindex < bindex){
            c.terms.insert(*ithis);
            ithis++;
        }
        else if(thisindex == bindex){                          //只有位置相同的
元素才相加，其他不变
            if((*ithis).value+(*ib).value != 0){
                matrixterm<T> temp;
                temp.row=(*ithis).row;
                temp.col=(*ithis).col;
```

```
                temp.value=(*ithis).value+(*ib).value;
                c.terms.insert(temp);
            }
            ithis++;
            ib++;
        }
        else{
            c.terms.insert(*ib);
            ib++;
        }
    }
    while(ithis!=terms.end()){
        c.terms.insert(*ithis);
        ithis++;
    }
    while(ib!=b.terms.end()){
        c.terms.insert(*ib);
        ib++;
    }
    this->copy(c);
}

template<class T>
void mysparsematrix<T>::multipul(mysparsematrix<T> &b){
    if(cols!=b.rows){
        cout<<-1<<endl;
        this->copy(b);
        return ;
    }
    mysparsematrix<T>c;
    c.rows = rows;
    c.cols = b.cols;
    c.terms.clear();
    b.transpose();                  //转置之后方便操作
    typename myarray<matrixterm<T>>::iterator ithis = terms.begin();
    typename myarray<matrixterm<T>>::iterator ib = b.terms.begin();
    typename myarray<matrixterm<T>>::iterator p = terms.begin();
    for(int i=1;i<=rows;i++){
        p = ithis;                                  //标记每一排的起始元组
        for(int j=1;j<=b.rows;j++){
            T sum=0;
            while(ithis!=terms.end() && ib!=b.terms.end() && ithis->row==i &&
ib->row==j){
                if(ithis->col==ib->col){                        //列相等的是对应
的位子
                    sum+=ithis->value * ib->value;          //乘完加到 sum 里
```

```
                    ithis++;
                    ib++;
                }
                else if(ithis->col < ib->col){
                    ithis++;
                }
                else {
                    ib++;
                }
            }
            if(sum!=0){
                matrixterm<T> temp;
                temp.row=i;
                temp.col=j;
                temp.value=sum;
                c.terms.insert(temp);
            }
            if(j!=b.rows) ithis = p;        //一排乘完，ithis 回到本行行首
        }
        ib = b.terms.begin();     //i 一轮走完，要换下一排了，ib 回到 b 矩阵的开头
    }
    this->copy(c);
}

template<class T>
void mysparsematrix<T>::output(){        //输出矩阵
    cout<<rows<<' '<<cols<<endl;
    typename myarray<matrixterm<T>>::iterator ithis = terms.begin();
    for(int i=1;i<=rows;i++){
        for(int j=1;j<=cols;j++){
            if(ithis!=terms.end() && (*ithis).row==i && (*ithis).col==j){
                cout<<(*ithis).value<<' ';
                ithis++;
            }
            else{
                cout<<0<<' ';
            }
        }
        cout<<endl;
    }
}

int main(){
    int n;
    cin>>n;
    mysparsematrix<int>P;
```

```
    for(int i=0;i<n;i++){
        int p;
        cin>>p;
        if(p==1){
            P.reset();
        }
        else if(p==2){
            mysparsematrix<int>b;
            b.set();
            P.multipul(b);
        }
        else if(p==3){
            mysparsematrix<int>b;
            b.set();
            P.add(b);
        }
        else if(p==4){
            P.output();
        }
        else if(p==5){
            P.transpose();
        }
    }
}
```