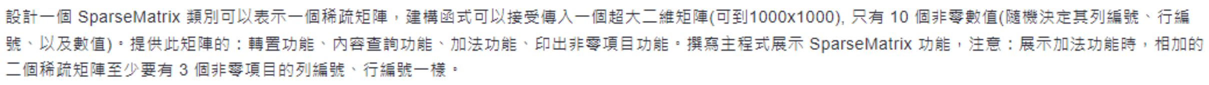
12/22週作業


設計一個 SparseMatrix 類別可以表示一個稀疏矩陣，建構函式可以接受傳入一個超大二維矩陣(可到1000x1000), 只有 10 個非零數值(隨機決定其列編號、行編號、以及數值)。提供此矩陣的：轉置功能、內容查詢功能、加法功能、印出非零項目功能。撰寫主程式展示 SparseMatrix 功能，注意：展示加法功能時，相加的二個稀疏矩陣至少要有 3 個非零項目的列編號、行編號一樣。

```
/* 12/22 楊育哲
 * 實作稀疏矩陣
 */
public class test {
    public static void main(String args[]){
        // show constructor using 1000*1000 array
        System.out.println("show constructor using 1000*1000 
        int[][] array = new int[1000][1000];
        for(int i=0; i<10; i++){
            int x=(int)(Math.random()*999), y=(int)(Math.rand
            array[x][y] = v;
        }
        SparseMatrix test = new SparseMatrix(array);
        test.view();

        // show add function
        System.out.println("show add function");
        int[][] constArray1=new int[3][1];
        int[][] constArray2=new int[3][1];
        for(int i=0; i<3; i++){
            constArray1[i][0]=(int)(Math.random()*98+1); cons
        }
        SparseMatrix test2 = new SparseMatrix(constArray1);
        SparseMatrix test3 = new SparseMatrix(constArray2);
        // !-------------------------------------------------
        // 看未重疊位置的加法
        test2.push((int)(Math.random()*96)+4, (int)(Math.rand
        test3.push((int)(Math.random()*96)+4, (int)(Math.rand
        // !-------------------------------------------------
        System.out.println("view const array1 values, and it'
        test2.view();
        System.out.println("view const array2 values, and it'
        test3.view();
        SparseMatrix afterAdd = test2.add(test3);
        System.out.println("after add:");
        afterAdd.view();

        //show trandform
        System.out.println("show transform function\nbefore:"
        test.view();
        test.transfrom("right");
        System.out.println("after transfrom right");
        test.view();

        //show lookfor
        System.out.println("show lookfor function\n lookfor (
        System.out.println("-1 repersent didn.t have value li
    }
}
class SparseMatrix{
    private int[][] sm; // = new int[10][3]
    private int capacity, items;
    private int r, c;
    SparseMatrix(){
        capacity=2; items=0; r=0; c=0;
        sm = new int[capacity][3];
    }
    SparseMatrix(int[][] init){
        capacity=2; items=0; r=0; c=0;
        sm = new int[capacity][3];
        for(int i=0; i<init.length; i++){
            int[] temp=init[i];
            for(int j=0; j<temp.length; j++){
                if(init[i][j]!=0) push(i, j, init[i][j]);
            }
        }
    }
    public void push(int x, int y, int v){
        if(items==capacity-1){
            int[][] tmp = new int[capacity*2][3];
            for(int k=0; k<capacity; k++){
                tmp[k][0] = sm[k][0]; //x
                tmp[k][1] = sm[k][1]; //y
                tmp[k][2] = sm[k][2]; //value
            }
            capacity*=2; sm=null;
            sm = tmp;
        }
        sm[items][0]=x; r=Math.max(x+1, r);
        sm[items][1]=y; c=Math.max(y+1, c);
        sm[items++][2]=v;
    }
    public void transfrom(String dir){
        if(dir.charAt(0)=='L'||dir.charAt(0)=='l'){
            for(int i=0; i<items; i++){
                int oldX=sm[i][0];
                sm[i][0]=sm[i][1]; //new_x = old_y
                sm[i][1]=r-1-oldX; //new_y = r-1-old_x
            }
        }else if(dir.charAt(0)=='R'||dir.charAt(0)=='r'){
            for(int i=0; i<items; i++){
                int oldY=sm[i][1];
                sm[i][1]=sm[i][0]; //new_y = old_x
                sm[i][0]=c-1-oldY; //new_y = r-1-old_x
            }
        }else System.out.println("input r or l");
    }
    public int lookfor(int x, int y){
        for(int i=0; i<items; i++){
            if(sm[i][0]==x&&sm[i][1]==y){
                return sm[i][2]-0;
            }
        }
        return -1;
    }
    public SparseMatrix add(SparseMatrix B){
        int i=0, j=0;
        SparseMatrix C = new SparseMatrix();
        while(i<items&&j<B.items){
            if(sm[i][0]==B.sm[j][0]&&sm[i][1]==B.sm[j][1]) C.
            else{
                C.push(sm[i][0], sm[i][1], sm[i][2]);
                C.push(B.sm[j][0], B.sm[j][1], B.sm[j][2]);
            }
            i++; j++;
        }
        while(i<items) C.push(sm[i][0], sm[i][1], sm[i][2]);
        while(j<B.items) C.push(B.sm[j][0], B.sm[j][1], B.sm[
        return C;
    }
    public void view(){
        for(int i=0; i<items; i++) System.out.println("["+sm[
    }
}
```

輸出:

```
show constructor using 1000*1000 array
[25, 143]=91
[39, 681]=4
[623, 998]=18
[645, 539]=43
[779, 523]=31
[780, 735]=39
[837, 510]=77
[859, 273]=93
[873, 967]=52
[970, 779]=89
show add function
view const array1 values, and it's random...:
[0, 0]=55
[1, 0]=6
[2, 0]=98
[15, 90]=123
view const array2 values, and it's random...:
[0, 0]=14
[1, 0]=35
```

```
[2, 0]=52
[96, 40]=456
after add:
[0, 0]=69
[1, 0]=41
[2, 0]=150
[15, 90]=123
[96, 40]=456
show transform function
before:
[25, 143]=91
[39, 681]=4
[623, 998]=18
[645, 539]=43
[779, 523]=31
[780, 735]=39
[837, 510]=77
[859, 273]=93
[873, 967]=52
[970, 779]=89
after transfrom right
[855, 25]=91
[317, 39]=4
[0, 623]=18
[459, 645]=43
[475, 779]=31
[263, 780]=39
[488, 837]=77
[725, 859]=93
[31, 873]=52
[219, 970]=89
show lookfor function
lookfor (0, 0) from test2(have)=55
-1 repersent didn.t have value like (4, 4)from test2: -1
```