题目 顺时针打印矩阵

考点 画图让抽象形象化 热点指数 58075 通过率 17.54%

具体题目

输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字，例如，如果输入如下4 X 4矩阵：

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

则依次打印出数字1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10.

```
import java.util.ArrayList;
public class Solution {
    ArrayList  a=new ArrayList();    new一个数组 以便下面函数能调用
    public ArrayList printMatrix(int [][] matrix) {
       int tR=0;
       int tC=0;
       int dR=matrix.length-1;
       int dC=matrix[0].length-1;
        while(tR<=dR&&tC<=dC){ 左上边界最多到达右下边界 用于判断是否还是剥圈打印
       printEdge(matrix,tR++,tC++,dR--,dC--);
        }
      return a;
    }
    public  void printEdge(int [][] m,int tR,int tC,int dR,int dC){
        if(tR==dR){     先判断是否只是一横行 如果是 打印该横行的列（通常用于内圈）
            for(int i=tC;i<=dC;i++){
                a.add(m[tR][i]);
            }
        }
        else if(tC==dC){   再判断是否只是一竖列 如果是 打印该横行的列（通常用于内圈
            for(int i=tR;i<=dR;i++){
                a.add(m[i][tC]);
            }
        }
        else {
            int curC=tC;用2个变量储存 用于判断当前位置
            int curR=tR;
            while(curC!=dC){       当前位置未到达当前行的最右列 --》往右去
                a.add(m[tR][curC]);
            curC++;
            }
            while(curR!=dR){       当前位置未到达当前列的最底行 --》往下去
                a.add(m[curR][dC]);
                curR++;
            }
            while(curC!=tC){       当前位置未到达当前行的最左列 --》往左去
                a.add(m[dR][curC]);
                curC--;
            }
            while(curR!=tR){       当前位置未到达当前列的最顶行 --》往上去
                a.add(m[curR][tC]);
```

```
                curR--;
            }
        }
    }
}
```

1.每次都是一个圈，所以定义四个变量限定每次循环的界限：
  startRow,endRow,startCol,endCol；
2.分别把首行，末列，末行，首列的数据依次加入list；
3.注意不要重复加入某个点，每次都要限定界限。
代码如下：

```
    public ArrayList<Integer> printMatrix(int [][] matrix) {
        int row = matrix.length;
        if(row==0)
            return null;
        int col = matrix[0].length;
        if(col==0)
            return null;
        ArrayList<Integer> list = new ArrayList<Integer>();

        int startRow = 0;
        int endRow = row-1;
        int startCol = 0;
        int endCol = col-1;
        while(startRow<=endRow&&startCol<=endCol){
            //如果就剩下一行
            if(startRow==endRow){
                for(int i=startCol;i<=endCol;i++)
                    list.add(matrix[startRow][i]);
                return list;
            }
            //如果就剩下一列
            if(startCol==endCol){
                for(int i=startRow;i<=endRow;i++)
                    list.add(matrix[i][startCol]);
                return list;
            }
            //首行
            for(int i=startCol;i<=endCol;i++)
                list.add(matrix[startRow][i]);
            //末列
            for(int i=startRow+1;i<=endRow;i++)
                list.add(matrix[i][endCol]);
            //末行
            for(int i=endCol-1;i>=startCol;i--)
                list.add(matrix[endRow][i]);
            //首列
            for(int i=endRow-1;i>=startRow+1;i--)
                list.add(matrix[i][startCol]);

            startRow = startRow + 1;
            endRow = endRow - 1;
            startCol = startCol + 1;
            endCol = endCol - 1;
        }
        return list;
    }
```

```
/**
 * @description 顺时针打印矩阵
 * @author GongchuangSu
```

```
 * @since 2016.09.03
 * @explain 输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字，例如，如果输入如下矩阵：
 *                      1 2 3 4
 *                      5 6 7 8
 *                      9 10 11 12
 *                      13 14 15 16
 *           则依次打印出数字
 *                      1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10.
 */
import java.util.*;
public class Solution{
    ArrayList<Integer> list = new ArrayList<>();

    public ArrayList<Integer> printMatrix(int [][] matrix) {
        int rows = matrix.length;
        int columns = matrix[0].length;
        int start = 0;
        while(rows > start*2 && columns > start*2){
            printMatrixInCircle(matrix, rows, columns, start);
            start++;
        }
        return list;
    }

    /**
     * 功能：打印一圈
     */
    public void printMatrixInCircle(int [][] matrix, int rows, int columns, int start){
        // 从左到右打印一行
        for(int i = start; i < columns - start; i++)
            list.add(matrix[start][i]);
        // 从上到下打印一列
        for(int j = start + 1; j < rows - start; j++)
            list.add(matrix[j][columns - start - 1]);
        // 从右到左打印一行
        for(int m = columns - start - 2; m >= start && rows - start - 1 > start; m--)
            list.add(matrix[rows - start - 1][m]);
        // 从下到上打印一列
        for(int n = rows - start - 2; n >= start + 1 && columns - start - 1 > start; n--)
            list.add(matrix[n][start]);
    }
}
```

```
import java.util.ArrayList;
public class Solution {
    public ArrayList<Integer> printMatrix(int [][] matrix) {
       ArrayList<Integer> ls = new ArrayList<Integer>();
        int colStart = 0;
        int colEnd = matrix[0].length;
        int lineStart = 0;
        int lineEnd = matrix.length;
        int count = lineEnd * colEnd;
        if (matrix == null)
            return ls;
        while (count != 0) {
            for(int i = colStart;i<colEnd;i++){
                ls.add(matrix[lineStart][i]);
                count--;
            }
            lineStart++;
            if(count==0)
```

```
                break;
            for(int i = lineStart;i<lineEnd;i++){
                ls.add(matrix[i][colEnd-1]);
                count--;
            }
            colEnd--;
            if(count==0)
                break;
            for(int i = colEnd-1;i>=colStart;i--){
                ls.add(matrix[lineEnd-1][i]);
                count--;
            }
            lineEnd--;
            if(count==0)
                break;
            for(int i = lineEnd-1;i>=lineStart;i--){
                ls.add(matrix[i][colStart]);
                count--;
            }
            colStart++;
            if(count==0)
                break;
        }
        return ls;
    }
}
```