顺时针打印矩阵\_剑指Offer\_29

# 顺时针打印矩阵\_剑指Offer\_29

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*剑指Offer29：顺时针打印矩阵\*\*\*\*\*\*\*\*\*\*\*\*\*/

\* 题目描述

\* 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.

## 思路分析

\* 思路分析：每一圈的起点的坐标x和y都是相等的，即(start,start)

\* 每一圈分成四步骤：第一步打印上行；

\* 第二步打印右列；

\* 第三步打印下行；

\* 第四步打印左列。

## Java代码

### if判断每一行或列的打印

public ArrayList<Integer> **printMatrix**(int [][] matrix) {

ArrayList<Integer> result = new ArrayList<Integer>();

if(matrix == null || matrix.length == 0) return result;

int rows = matrix.length;

int columns = matrix[0].length;

/\*\*

\* 分析start的限制条件：由于起点是(start,start)，显然start由行数和列数的一半的较小值决定，

\* 当行数为奇数时，如rows=5，start需要0,1,2；当rows=4时，start需要0,1；因此row<(rows+1)>>1或者row<<1 <rows

\*/

for(int start = 0;start<<1 < rows && start<<1 < columns;start++){

int rowEnd = rows -1 -start;//下行行号

int columnEnd = columns -1 -start;//右列列号

//打印上行

for(int i = start;i <= columnEnd;i++){

result.add(matrix[start][i]);

}

//打印右列

if**(rowEnd > start) {** //至少需要两行

for (int i = start + 1; i <= rowEnd; i++) {

result.add(matrix[i][columnEnd]);

}

}

//打印下行：

if(**rowEnd>start&&columnEnd>start**) {//至少两行且至少两列才需要打印：

for (int i = columnEnd - 1; i >= start; i--) {

result.add(matrix[rowEnd][i]);

}

}

//打印左列

if(**rowEnd > start+1 &&columnEnd > start**) {//至少三行两列

for (int i = rowEnd - 1; i > start && columnEnd > start; i--) {

result.add(matrix[i][start]);

}

}

}

return result;

}

### 条件整合到for循环的条件判断中

/\*\*

\* 将外加if判断条件转入for循环的判断条件中；原理和上一方法一致

\*/

public ArrayList<Integer> printMatrix2(int [][] matrix) {

ArrayList<Integer> result = new ArrayList<Integer>();

if(matrix == null || matrix.length == 0) return result;

int rows = matrix.length;

int columns = matrix[0].length;

/\*\*

\* 分析start的限制条件：由于起点是(start,start)，显然start由行数和列数的一半的较小值决定，

\* 当行数为奇数时，如rows=5，start需要0,1,2；当rows=4时，start需要0,1；因此row<(rows+1)>>1或者row<<1 <rows

\*/

for(int start = 0;start<<1 < rows && start<<1 < columns;start++){

int rowEnd = rows -1 -start;//下行行号

int columnEnd = columns -1 -start;//右列列号

//打印上行

for(int i = start;i <= columnEnd;i++){

result.add(matrix[start][i]);

}

//打印右列:至少需要两行

// 从start+1开始至小于等于rowEnd限制了至少两行

for(int i = start+1;i <= rowEnd;i++){

result.add(matrix[i][columnEnd]);

}

//打印下行：//至少两行且至少两列才需要打印：

// ①rowEnd > start控制了至少两行；

// ②从columnEnd-1开始至大于等于start控制了至少两列

for(int i = columnEnd-1;i>= start&& rowEnd > start;i--){

result.add(matrix[rowEnd][i]);

}

//打印左列：至少三行两列:columnEnd > start控制了至少两列；从rowEnd-1开始至小于start控制了至少三行

for(int i = rowEnd-1;i>start&&columnEnd>start;i--){

result.add(matrix[i][start]);

}

}

return result;

}