# 题目

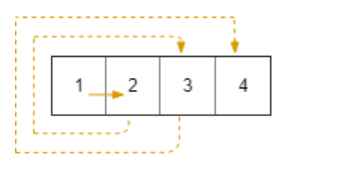
在 rows x cols 的网格上，你从单元格 (rStart, cStart) 面朝东面开始。网格的西北角位于第一行第一列，网格的东南角位于最后一行最后一列。

你需要以顺时针按螺旋状行走，访问此网格中的每个位置。每当移动到网格的边界之外时，需要继续在网格之外行走（但稍后可能会返回到网格边界）。

最终，我们到过网格的所有 rows x cols 个空间。

按照访问顺序返回表示网格位置的坐标列表。

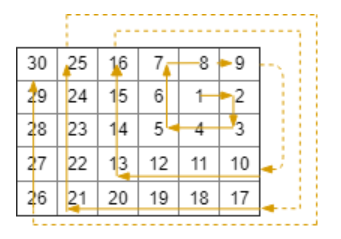
示例 1：



输入：rows = 1, cols = 4, rStart = 0, cStart = 0

输出：[[0,0],[0,1],[0,2],[0,3]]

示例 2：



输入：rows = 5, cols = 6, rStart = 1, cStart = 4

输出：[[1,4],[1,5],[2,5],[2,4],[2,3],[1,3],[0,3],[0,4],[0,5],[3,5],[3,4],[3,3],[3,2],[2,2],[1,2],[0,2],[4,5],[4,4],[4,3],[4,2],[4,1],[3,1],[2,1],[1,1],[0,1],[4,0],[3,0],[2,0],[1,0],[0,0]]

提示：

1 <= rows, cols <= 100

0 <= rStart < rows

0 <= cStart < cols

# 分析

## 方法一：模拟

思路：

1、使用一个二维数组 result 来存放结果，起始位置为 (rStart, cStart)，将其加入到 result 中。

2、使用 direction 来表示当前移动的方向，0 表示向东，1 表示向南，2 表示向西，3 表示向北。

3、使用 step 表示当前的步长，初始值为1，每经过一轮向四个方向移动结束后，增加步长。

4、在每个方向上移动 step 步，并将符合条件的坐标加入到 result 中。

5、不断重复以上步骤，直到 result 中包含了所有格子的坐标。

代码：

class Solution {

public:

vector<vector<int>> spiralMatrixIII(int rows, int cols, int rStart, int cStart) {

vector<vector<int>> result; // 用于存放结果的二维数组

int total = rows \* cols; // 总共需要遍历的格子数

int r = rStart, c = cStart; // 当前位置的行和列

int direction = 0; // 方向，0: east, 1: south, 2: west, 3: north

int step = 1; // 当前步长

result.push\_back({r, c}); // 将起始位置加入结果中

while (result.size() < total) { // 当结果中的坐标数量小于总格子数时继续遍历

// 向东移动

for (int i = 0; i < step; ++i) {

c++;

if (isValid(rows, cols, r, c)) { // 检查新位置是否在网格内

result.push\_back({r, c});

}

}

// 向南移动

for (int i = 0; i < step; ++i) {

r++;

if (isValid(rows, cols, r, c)) {

result.push\_back({r, c});

}

}

step++; // 增加步长

// 向西移动

for (int i = 0; i < step; ++i) {

c--;

if (isValid(rows, cols, r, c)) {

result.push\_back({r, c});

}

}

// 向北移动

for (int i = 0; i < step; ++i) {

r--;

if (isValid(rows, cols, r, c)) {

result.push\_back({r, c});

}

}

step++; // 增加步长

}

return result;

}

private:

// 检查坐标是否在网格内

bool isValid(int rows, int cols, int r, int c) {

return r >= 0 && r < rows && c >= 0 && c < cols;

}

};