# 题目

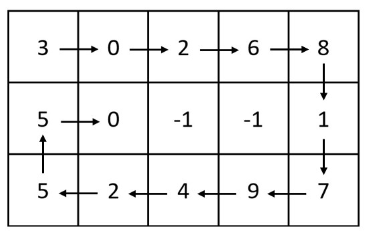
给你两个整数：m和n ，表示矩阵的维数。

另给你一个整数链表的头节点head 。

请你生成一个大小为m x n的螺旋矩阵，矩阵包含链表中的所有整数。链表中的整数从矩阵左上角开始、顺时针按螺旋顺序填充。如果还存在剩余的空格，则用-1填充。

返回生成的矩阵。

示例 1：



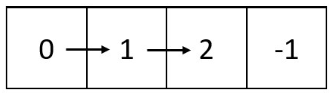
输入：m = 3, n = 5, head = [3,0,2,6,8,1,7,9,4,2,5,5,0]

输出：[[3,0,2,6,8],[5,0,-1,-1,1],[5,2,4,9,7]]

解释：上图展示了链表中的整数在矩阵中是如何排布的。

注意，矩阵中剩下的空格用 -1 填充。

示例 2：



输入：m = 1, n = 4, head = [0,1,2]

输出：[[0,1,2,-1]]

解释：上图展示了链表中的整数在矩阵中是如何从左到右排布的。

注意，矩阵中剩下的空格用 -1 填充。

提示：

1 <= m, n <= 105

1 <= m \* n <= 105

链表中节点数目在范围 [1, m \* n] 内

0 <= Node.val <= 1000

# 分析

## 方法一：模拟

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

// 四个方向，按右、下、左、上顺序记录（因为螺旋也是按这个顺序转的）

short dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};

public:

vector<vector<int>> spiralMatrix(int n, int m, ListNode\* head) {

// 初始化答案矩阵，并将所有位置设为 -1

vector<vector<int>> ans;

ans.resize(n, vector<int>(m));

for (int i = 0; i < n; i++) for (int j = 0; j < m; j++) ans[i][j] = -1;

// (i, j) 是当前矩阵中的坐标，D 是当前的方向

int i = 0, j = 0, D = 0;

while (true) {

// 填写当前位置

ans[i][j] = head->val;

head = head->next;

// 已经没有下一个位置要填了，退出

if (head == nullptr) break;

// 算出下一个位置的坐标

while (true) {

// 试着往当前方向走一步

int ii = i + dir[D][0], jj = j + dir[D][1];

// 会走出矩阵，换一个方向

if (ii < 0 || jj < 0 || ii >= n || jj >= m) D = (D + 1) % 4;

else if (ans[ii][jj] >= 0) { // 走到已有数字的格子中，换一个方向

D = (D + 1) % 4;

} else {

// 走到下一个格子

i = ii;

j = jj;

break;

}

}

}

return ans;

}

};