1. Spiral Matrix

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

**Example 1:**

Input:  
[  
 [ 1, 2, 3 ],  
 [ 4, 5, 6 ],  
 [ 7, 8, 9 ]  
]  
Output: [1,2,3,6,9,8,7,4,5]

**Example 2:**

Input:  
[  
 [1, 2, 3, 4],  
 [5, 6, 7, 8],  
 [9,10,11,12]  
]  
Output: [1,2,3,4,8,12,11,10,9,5,6,7]

**解**

解法1 模拟，想像一个小车在地图上遍历

class Solution {  
public:  
 vector<int> spiralOrder(vector<vector<int>>& matrix) {  
 vector<int>ans;  
 if(matrix.size() == 0)return ans;  
 int x = 0, y = 0;  
 int Xs = 0, Ys = 0;  
 int Xe = matrix[0].size() - 1, Ye = matrix.size() - 1;  
 int dx = 1, dy = 0;  
 int direction = 1; // 1: right, 2:left, 3:up, 4 : down  
 while(Xs <= Xe && Ys <= Ye){  
 ans.push\_back(matrix[y][x]);  
 int tmpX = x + dx, tmpY = y + dy;  
 if(tmpX > Xe || tmpX < Xs || tmpY > Ye || tmpY < Ys){  
 switch(direction){  
 case 1 :  
 direction = 4;  
 dx = 0, dy = 1;  
 Ys++;  
 break;  
 case 2:  
 direction = 3;  
 dx = 0, dy = -1;  
 Ye--;  
 break;  
 case 3 :  
 direction = 1;  
 dx = 1, dy = 0;  
 Xs++;  
 break;  
 case 4 :  
 direction = 2;  
 dx = -1, dy = 0;  
 Xe--;  
 break;  
 }  
 }  
 x += dx; y += dy;  
 }  
 return ans;  
 }  
};

解法2 逐层存。注意在存完相邻的两个边之后，要进行判断，防止重复存

class Solution {  
public:  
 vector<int> spiralOrder(vector<vector<int>>& matrix) {  
 vector<int>ans;  
 if(matrix.size() == 0)return ans;  
 int Xs = 0, Ys = 0;  
 int Xe = matrix[0].size() - 1, Ye = matrix.size() - 1;  
 while(Xs <= Xe && Ys <= Ye){  
 for(int i = Xs; i <= Xe; ++i)ans.push\_back(matrix[Ys][i]);  
 for(int i = Ys + 1; i <= Ye - 1; ++i)ans.push\_back(matrix[i][Xe]);  
 for(int i = Xe ; i >= Xs && Ys != Ye; --i)ans.push\_back(matrix[Ye][i]);  
 for(int i = Ye - 1; i >= Ys + 1 && Xs != Xe; --i)ans.push\_back(matrix[i][Xs]);  
 Xs++, Xe--;  
 Ys++, Ye--;  
 }  
 return ans;  
 }  
};