画解剑指 Offer - LeetBook - 力扣（LeetCode）全球极客挚爱的技术成长平台

剑指 Offer 29. 顺时针打印矩阵 - 解决方案

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。

示例 1：

```
输入：matrix = [[1,2,3],[4,5,6],[7,8,9]]
输出：[1,2,3,6,9,8,7,4,5]
```

示例 2：

```
输入：matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
输出：[1,2,3,4,8,12,11,10,9,5,6,7]
```

限制：

- `0 <= matrix.length <= 100`
- `0 <= matrix[i].length <= 100`

注意：本题与主站 54 题 相同

解题方案

思路

- 标签：二维数组
- 整体思路：循环遍历整个数组，循环中再嵌套四个循环，分别是从左至右，从上至下，从右至左，从下至上这几个方向，按照题意将整个数组遍历完成，控制好边界
- mm 为行数，nn 为列数，时间复杂度：O(mn)O(mn)，空间复杂度：O(1)O(1)

算法流程

1. 题目中 `matrix` 有可能为空，直接返回空数组即可
2. 初始化边界 `left` 、 `right` 、 `top` 、 `bottom` 四个值，初始化结果数组 `res` 和数组下标 `x`
3. 按照遍历方向循环取出数字放入结果数组中

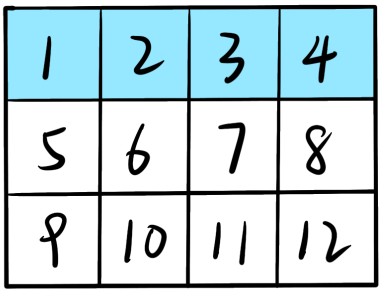
- 从左至右：遍历完成后 `++top` ，如果 `top > bottom` ，到达边界循环结束
- 从上至下：遍历完成后 `--right` ，如果 `left > right` ，到达边界循环结束

- 从右至左：遍历完成后 `--bottom` ，如果 `top > bottom` ，到达边界循环结束
- 从下至上：遍历完成后 `++left` ，如果 `left > right` ，到达边界循环结束

代码

```
class Solution {
    public int[] spiralOrder(int[][] matrix) {
        if(matrix.length == 0) return new int[0];
        int left = 0, right = matrix[0].length - 1, top = 0, bottom =
matrix.length - 1, x = 0;
        int[] res = new int[(right + 1) * (bottom + 1)];
        while(true) {
            for(int i = left; i <= right; i++) res[x++] = matrix[top][i];
            if(++top > bottom) break;
            for(int i = top; i <= bottom; i++) res[x++] = matrix[i][right];
            if(left > --right) break;
            for(int i = right; i >= left; i--) res[x++] = matrix[bottom][i];
            if(top > --bottom) break;
            for(int i = bottom; i >= top; i--) res[x++] = matrix[i][left];
            if(++left > right) break;
        }
        return res;
    }
}
```

画解



1 / 6

花絮

讨论区
共 10 个讨论

最热

匿名用户

来自山东2021-12-20

代码精简挺好的，但是我当时我第一眼看代码的时候，我都不知道哪跟哪！在我看了其他板块的代码之后，才知道你的代码是这么回事。希望还是不要省略步骤，这样真的对新手不友好。我刚才差点就放弃了！

2

Mustard L1
来自河南2021-03-05

贴个C++

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
    vector<int> result;
    if(matrix.empty()) return result;
    int top = 0;int button = matrix.size();int left = 0;int right =
matrix[0].size();


    while(true) {
        //按顺序循环
        for(int i = left;i< right;++i) result.push_back(matrix[top][i]);
        if(++top >= button) break;
        for(int i = top;i <button;++i) result.push_back(matrix[i][right-1]);
        if(--right <= left) break;
        for(int i = right-1;i >= left;--i) result.push_back(matrix[button-1][i]);
        if(--button <= top)break;
        for(int i = button-1; i >= top;--i) result.push_back(matrix[i][left]);
        if(++left >= right) break;
    }
    return result;
    }
};
```

3

乐者不玉 L2
来自美国2021-05-14

JS来了，while中的switch case来判断方向

```
/**
 * @param {number[][]} matrix
 * @return {number[]}
 */
var spiralOrder = function (matrix) {
  if (!matrix.length) {
    return [];
  }
  let top = 0,
    left = 0,
    bottom = matrix.length - 1,
    right = matrix[0].length - 1,
    direction = "right",
    result = [];

  while (left <= right && top <= bottom) {
    switch (direction) {
      case "right": {
        for (let i = left; i <= right; i++) {
          result.push(matrix[top][i]);
        }
        top++;
        direction = "down";
        break;
      }
      case "down": {
        for (let i = top; i <= bottom; i++) {
          result.push(matrix[i][right]);
        }
        right--;
        direction = "left";
        break;
      }
      case "left": {
        for (let i = right; i >= left; i--) {
          result.push(matrix[bottom][i]);
        }
        bottom--;
        direction = "top";
        break;
      }
      case "top": {
        for (let i = bottom; i >= top; i--) {
          result.push(matrix[i][left]);
        }
        left++;
        direction = "right";
        break;
      }
      default: {
        break;
      }
    }
  }
  return result;
```

```
};
```

2

已注销

来自天津2021-03-08

贴个js

```
/**
 * @param {number[][]} matrix
 * @return {number[]}
 */
/**
 * [1, 2, 3]
 * [4, 5, 6]
 * [7, 8, 9]
 *
 * [ 1,  2,  3,  4]
 * [ 5,  6,  7,  8]
 * [ 9, 10, 11, 12]
 *
 */
var spiralOrder = function(matrix) {
    let row = 0, col = 0;
    let direction = "right", border = 0;    //direction是当前遍历方向， border是边界
    let sum = 0, n = 0;                     //sum是矩阵中元素总数， n是当前遍历的元素数
量
    let arr = [];                           //返回的数组
    for(let i = 0; i < matrix.length; ++i)
    {
        sum += matrix[i].length;
    }
    while(n < sum)
    {
        if(direction == 'right')
        {
            if(col <= matrix[0].length - 1 - border)
            {
                arr.push(matrix[row][col++]);
                ++n;
            }
            else
            {
                --col;
                ++row;
                direction = 'down';
            }
        }
        else if(direction == 'down')
        {
            if(row <= matrix.length - 1 - border)
            {
                arr.push(matrix[row++][col]);
                ++n;
            }
            else
            {
                --row;
                --col;
                direction = 'left';
            }
        }
        else if(direction == 'left')
        {
```

```
            if(col >= 0 + border)
            {
                arr.push(matrix[row][col--]);
                ++n;
            }
            else
            {
                ++col;
                --row;
                direction = 'up';
                ++border;
            }
        }
        else if(direction == 'up')
        {
            if(row >= 0 + border)
            {
                arr.push(matrix[row--][col]);
                ++n;
            }
            else
            {
                ++row;
                ++col;
                direction = 'right';
            }
        }
    }
    return arr;
};
```

1

秀川L1
来自北京2021-12-08

和作者的不太一样 我是添加一个递加的标识 表示转圈的次数来控制循环的条件

```
class Solution {
    public int[] spiralOrder(int[][] matrix) {
        if(matrix.length==0){
            return new int[0];
        }
        int target = 0;
        int row = 0;
        int col = 0;
        int[] res = new int[matrix.length * matrix[0].length];
        for(int i = 0;i<res.length;i++){
            res[i] = matrix[row][col];
            // 如果row==target&&col<matrix[0].length-1-target col++ 一直到
col==matrix[0].length-1-target
            // 如果col==matrix[0].length-1-target&&row<matrix.length-1-target row++
一直到row==matrix.length-1-target
            // 如果ow==matrix.length-1-target&&col>target 如果col-- 一直到
col==target
            // 如果col==target&&row>target row-- 一直到row=target
            if(row==target&&col<matrix[0].length-1-target){
                col++;
                continue;
            }
            if(col==matrix[0].length-1-target&&row<matrix.length-1-target){
                row++;
                continue;
            }
            if(row==matrix.length-1-target&&col>target){
                col--;
                continue;
            }
            if(row==target+1){
                target++;
                col++;
                continue;
            }
            if(col==target&&row>=target+1){
                row--;
                continue;
            }
        }
        return res;

    }
}
```

Sumauto

L2

来自江苏2021-12-07

「代码块」

```
class Solution {
    public int[] spiralOrder(int[][] matrix) {
        int row=matrix.length;
        int col;
        if(row==0||(col=matrix[0].length)==0)return new int[0];
        int nums[]=new int[row*col];
        int x=-1,y=0,dir=0;
        int l=0,t=0,r=col-1,b=row-1;
        for(int i=0;i<nums.length;i++){
            if(dir==0){//right
                if(x<r)x++;
                else{
                    y++;dir=1;t++;
                }
            }else if(dir==1){//b
                if(y<b)y++;
                else{
                    x--;dir=2;r--;
                }
            }else if(dir==2){//l
                if(x>l)x--;
                else{
                    y--;dir=3;b--;
                }
            }else if(dir==3){//t
                if(y>t)y--;
                else{
                    x++;dir=0;l++;
                }
            }
            nums[i]=matrix[y][x];
        }
        return nums;

    }
}
```

忍野扇 L1

来自广东2021-06-05

【C++】

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int m = matrix.size(), n = matrix[0].size();
        int rowUp = 0, rowDown = m-1, lineLeft = 0, lineRight = n-1, i = 0, j = 0;
        vector<int> ans;
        while(rowUp<=rowDown || lineLeft<=lineRight){
            while(rowUp<=rowDown && j<=lineRight){
                ans.push_back(matrix[i][j]);
                j++;
            }
            j--;
            i++;
            rowUp++;
            while(lineLeft<=lineRight && i<=rowDown){
                ans.push_back(matrix[i][j]);
                i++;
            }
            i--;
            j--;
            lineRight--;
            while(rowUp<=rowDown && j>=lineLeft){
                ans.push_back(matrix[i][j]);
                j--;
            }
            i--;
            j++;
            rowDown--;
            while(lineLeft<=lineRight && i>=rowUp){
                ans.push_back(matrix[i][j]);
                i--;
            }
            i++;
            j++;
            lineLeft++;
        }
        return ans;
    }
};
```

小宋啊 L3

来自北京2021-04-13

```
public class Solution {
    private int[] res;

    public int[] spiralOrder(int[][] matrix) {
        // 注意:  int[][] matrix = {{}}
        // matrix.length=1,matrix[0].length=0
        // 输入空列，返回空数组
        if (matrix.length == 0) {
            return new int[0];
        }
        // 初始化左上角、右下角坐标
        int tR = 0, tC = 0;
        int dR = matrix.length - 1, dC = matrix[0].length - 1;
        res = new int[matrix.length * matrix[0].length];
        int index = 0;
        while (tR <= dR && tC <= dC) {
            index = spiralMatrix(matrix, index, tR++, tC++, dR--, dC--);
        }
        return res;
    }

    private int spiralMatrix(int[][] matrix, int index, int tR, int tC, int dR,
int dC) {
        if (tR == dR) {// 子矩阵只有一行
            for (int i = tC; i <= dC; i++) {
                res[index++] = matrix[tR][i];
            }
        } else if (tC == dC) {// 子矩阵只有一列
            for (int i = tR; i <= dR; i++) {
                res[index++] = matrix[i][tC];
            }
        } else {
            int curR = tR;
            int curC = tC;
            while (curC != dC) {
                res[index++] = matrix[tR][curC++];
            }
            while (curR != dR) {
                res[index++] = matrix[curR++][dC];
            }
            while (curC != tC) {
                res[index++] = matrix[dR][curC--];
            }
            while (curR != tR) {
                res[index++] = matrix[curR--][tC];
            }
        }
        return index;
    }

    public static void main(String[] args) {
        Solution solution = new Solution();
        int[][] matrix = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14,
15, 16}};
//        int[][] matrix = {{}};
//        System.out.println(matrix.length);
```

```
//        System.out.println(matrix[0].length);
        int[] res = solution.spiralOrder(matrix);
        System.out.println(Arrays.toString(res));
    }
}
```

👍

Knight

来自辽宁2021-04-06

```
class Solution {
    public int[] spiralOrder(int[][] matrix) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0){
            int[] a=new int[]{};
             return a;
        }

        int l = 0;
        int r = matrix[0].length;
        int u = 0;
        int d = matrix.length;
        int m=0;
        int[] array=new int[r*d];
        r--;
        d--;
        while (l <= r && u <= d){
            for (int i = l; i <= r; i++) {
                array[m]=matrix[u][i];
                m++;
            }
            u++;
            for (int i = u; i <= d; i++) {
                array[m]=matrix[i][r];
                m++;
            }
            r--;
            for (int i = r; i >= l && u <= d; i--) {
                array[m]=matrix[d][i];
                m++;
            }
            d--;
            for (int i = d; i >= u && l <= r; i--) {
                array[m]=matrix[i][l];
                m++;
            }
            l++;
        }
        return array;
    }
}
```

今天不刷题明天变废物

来自北京2021-03-08

```
class Solution:
    def spiralOrder(self, matrix: List[List[int]]) -> List[int]:
        help = []
        if matrix == []:
            return []
        left, right, top, buttom = 0, len(matrix[0]) - 1, 0, len(matrix) - 1
        while True:
            for i in range(left, right + 1):
                help.append(matrix[top][i])
            top += 1
            if top > buttom:
                break
            for i in range(top, buttom + 1):
                help.append(matrix[i][right])
            right -= 1
            if right < left:
                break
            for i in range(right, left - 1, -1):
                help.append(matrix[buttom][i])
            buttom -= 1
            if buttom < top:
                break
            for i in range(buttom, top - 1, -1):
                help.append(matrix[i][left])
            left += 1
            if left > right:
                break
        return help
```