**LC#54:SPIRAL MATRIX:**

**Example 1:**



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

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

**ONLY APPROACH:**

**-> iterate from left to right (top++)**

**-> iterate from top right to top bottom(right--)**

**->iterate from bottom right to bottom left(bottom--)**

**->iterate from bottom left to top left (left++)**

**HINT:**

**First write the path of flow**

**1)left->right**

**2)top->bottom**

**3)right->left**

**4)bottom->top**

**->the next path start is the previous one constant**

**->for 3)check top<=bottom**

**->for 4) check left<=right**

class Solution {

public List<Integer> spiralOrder(int[][] m) {

int top = 0;

int right = m[0].length-1;

int left = 0;

int bottom = m.length-1;

List<Integer> res = new ArrayList();

while(left<=right && top<=bottom)

{

**for(int i = left; i<=right;i++)//left to right**

**res.add(m[top][i]);**

**top++;**

**for(int i= top; i<=bottom;i++)//right corner to bottom right**

**res.add(m[i][right]);**

**right--;**

**if(top<=bottom)**

**{**

**for(int i = right; i >= left; i--)**

**res.add(m[bottom][i]);**

**bottom--;**

**}**

**if(left<=right){**

**for(int i = bottom ; i>=top;i--)**

**res.add(m[i][left]);**

**left++;**

}

}

return res;}}

**lc#59 SPIRAL MATRIX II**

**->same as spiral 1 but create a matrix and assign the val++to it (initial value of val=1)**

class Solution {

    public int[][] generateMatrix(int n) {

        int left = 0 ;

        int top = 0 ;

        int right = n-1;

        int bottom = n-1;

        int val = 1;

        int res[][]= new int[n][n];

        while(top<=bottom && left<=right)

        {

            for(int i = left ; i<=right;i++)

            {

                res[top][i] = val++;

            }

            top++;

            for(int i = top ; i<=bottom ;i++)

            {

                res[i][right] = val++;

            }

            right--;

            if(top<=bottom){

            for(int i = right ; i>=left ;i--)

            {

                res[bottom][i] = val++;

            }

            bottom--;

            }

            if(left<=right){

            for(int i = bottom;i>=top;i--)

            {

                res[i][left] = val++;

            }

            left++;

            }

        }

        return res;

    }

}

**LC#2326 SPIRAL MATRIX IV:**

**LINKED LIST**

**->same as spiral matrix but here linked list value is used**

**APPROACH:**

**->first fill the resultant array with -1**

**->do spiral turns**

**1)left -> right**

**2)top->bottom**

**3)right->left**

**4)bottom->top**

**The only thing here changes is**

**If(head!=null)**

**{**

**res[][] = head.val;**

**head = head.next;**

**}**

**->then others the same as spiral**

**CODE:**

/\*\*

 \* Definition for singly-linked list.

 \* public class ListNode {

 \*     int val;

 \*     ListNode next;

 \*     ListNode() {}

 \*     ListNode(int val) { this.val = val; }

 \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }

 \* }

 \*/

class Solution {

    public int[][] spiralMatrix(int m, int n, ListNode head) {

        int res[][] = new int[m][n];

        Arrays.stream(res).forEach(row->Arrays.fill(row,-1));

        int top = 0 , left = 0;

        int bottom = m-1 , right = n-1;

        ListNode temp = head;

        while(top<=bottom && left<=right)

        {

            for(int i = left ;i<=right ;i++)

            {

                if(head!=null)

                {

                    res[top][i] = head.val;

                    head = head.next;

                }

            }

            top++;

            for(int i = top;i<=bottom;i++)

            {

                if(head!=null)

                {

                    res[i][right] = head.val;

                    head = head.next;

                }

            }

            right--;

            if(top<=bottom)

            {

                for(int i = right ;i>=left;i--)

                {

                    if(head!=null)

                    {

                        res[bottom][i] = head.val;

                        head = head.next;

                    }

                }

                bottom--;

            }

            if(left<=right)

            {

                for(int i = bottom ; i>=top;i--)

                {

                    if(head!=null)

                    {

                        res[i][left] = head.val;

                        head = head.next;

                    }

                }

                left++;

            }

        }

        return res;

    }

}

**SPIRAL MATRIX III**

**HERE THEY HAVE GIVEN THE ROW AND COL AND THE STARTROW AND COL POSITION SUCH THAT SPIRAL REGRARDLESS OF BOUND UNTIL ALL COVERED**

class Solution {

    public int[][] spiralMatrixIII(int rows, int cols, int r, int c) {

        int total = rows \* cols;

        int[][] res = new int[total][2];

        int[][] dirs = {{0,1},{1,0},{0,-1},{-1,0}}; // right, down, left, up

        int count = 0, d = 0, steps = 1;

        res[count++] = new int[]{r, c};

**while(count < total) {**

**for(int i = 0; i < 2; i++) { // 2 directions before increasing steps**

**for(int j = 0; j < steps; j++) {**

                    r += dirs[d][0];

                    c += dirs[d][1];

                    if(r >= 0 && r < rows && c >= 0 && c < cols) {

                        res[count++] = new int[]{r, c};

                        if(count == total) return res;

                    }

                }

**d = (d + 1) % 4; // change direction**

            }

**steps++; // increase steps after 2 directions**

        }

        return res;

    }

}