**717821E113**

**2D array**

Q1. Spiral Matrix II Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order

class Solution {

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

        int top=0;

        int bottom=n-1;

        int left=0;

        int right=n-1;

        int[][] arr=new int[n][n];

        int val=1;

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

        //left to right

        for(int i=left;i<=right;i++){

            arr[top][i]=val;

            val++;

        }

        top++;

        //top to bottom

        for(int i=top;i<=bottom;i++){

            arr[i][right]=val;

            val++;

        }

        right--;

        //right to left

        for(int i=right;i>=left;i--){

            arr[bottom][i]=val;

            val++;

        }

        bottom--;

        //bottom to top

        for(int i=bottom;i>=top;i--){

            arr[i][left]=val;

            val++;

        }

        left++;

        }

        return arr;

    }

}

Q2. Richest Customer Wealth You are given an m x n integer grid accounts where accounts[i][j] is the amount of money the ith customer has in the jth bank. Return the wealth that the richest customer has. A customer's wealth is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum wealth

class Solution {

    public int maximumWealth(int[][] accounts) {

        int sum=0,max=0;

        for(int i=0;i<accounts.length;i++){

            for(int j=0;j<accounts[0].length;j++){

                sum+=accounts[i][j];

            }

            if(sum>=max)max=sum;

            sum=0;

        }

        return max;

    }

}

Q3. Toeplitz Matrix Given an m x n matrix, return true if the matrix is Toeplitz. Otherwise, return false. A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same elements.

class Solution {

    public boolean isToeplitzMatrix(int[][] matrix) {

        for(int i=1;i<matrix.length;i++){

            for(int j=1;j<matrix[0].length;j++){

                if(matrix[i][j]!=matrix[i-1][j-1]){

                    return false;

                }

            }

        }

        return true;

    }

}

Q4. Matrix Diagonal Sum Given a square matrix mat, return the sum of the matrix diagonals. Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

class Solution {

    public int diagonalSum(int[][] mat) {

        int row=mat.length;

        int col=mat[0].length;

        int pd=0;

        int sd=0;

        for(int i=0;i<row;i++){

             pd+=mat[i][i];}

         for(int i=col-1,j=0;i>=0;j++,i--){

             sd+=mat[j][i];}

       if(col%2!=0){

           int mid=mat[(row/2)][(col/2)];

            return pd+sd-mid;

       }else{

           return pd+sd;

       }

    }

}

Q5. Count Negative Numbers in a Sorted Matrix Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, return the number of negative numbers in grid.

class Solution {

    public int countNegatives(int[][] grid) {

        int count=0;

        for(int i=0;i<grid.length;i++){

            for(int j=0;j<grid[0].length;j++){

                if(grid[i][j]<0){

                    count+=grid[0].length-j;

                    break;

                }

            }

        }

        return count;

    }

}

Q6. Transpose Matrix Given a 2D integer array matrix, return the transpose of matrix. The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

class Solution {

    public int[][] transpose(int[][] matrix) {

        int rows=matrix.length;

        int cols=matrix[0].length;

        int[][] m=new int[cols][rows];

        for(int i=0;i<rows;i++){

            for(int j=0;j<cols;j++){

             m[j][i]=matrix[i][j];

            }

        }

        return m;

    }

}

Q7. Set Matrix Zeroes Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's.

class Solution {

    public void setZeroes(int[][] matrix) {

        int row=matrix.length-1;

        int col=matrix[0].length-1;

        //bruteforce

        //     for(int i=0;i<=row;i++){

        //         for(int j=0;j<=col;j++){

        //             if(matrix[i][j]==0){

        //                 markRow(i,matrix,col);

        //                 markCol(j,matrix,row);

        //             }

        //         }

        //     }

        //      for(int i=0;i<=row;i++){

        //         for(int j=0;j<=col;j++){

        //             if(matrix[i][j]==-10230){

        //                 matrix[i][j]=0;

        //             }

        //         }

        //     }

        // }

        //  public void markRow(int i,int[][] matrix,int col){

        //         for(int j=0;j<=col;j++){

        //             if(matrix[i][j]!=0){

        //                 matrix[i][j]= -10230;

        //             }

        //         }

        //     }

        //     public void markCol(int j,int[][] matrix,int row){

        //         for(int i=0;i<=row;i++){

        //             if(matrix[i][j]!=0){

        //                 matrix[i][j]= -10230;

        //             }

        //         }

        //better

        // int[] col1=new int[col+1];

        // int[] row1=new int[row+1];

        // for(int i=0;i<=row;i++){

        //     for(int j=0;j<=col;j++){

        //         if(matrix[i][j]==0){

        //             row1[i]=1;

        //             col1[j]=1;

        //         }

        //     }

        // }

        //  for(int i=0;i<=row;i++){

        //     for(int j=0;j<=col;j++){

        //         if(row1[i]==1||col1[j]==1){

        //             matrix[i][j]=0;

        //         }

        //     }

        // }

        //optimal solution

        int col0=1;

        for(int i=0;i<=row;i++){

            for(int j=0;j<=col;j++){

                if(matrix[i][j]==0){

                    matrix[i][0]=0;

                    if(j!=0){

                    matrix[0][j]=0;

                    }

                    else{

                    col0=0

                    }

                }

            }

        }

         for(int i=1;i<=row;i++){

            for(int j=1;j<=col;j++){

                if(matrix[i][j]!=0){

                    if(matrix[i][0]==0||matrix[0][j]==0){

                        matrix[i][j]=0;

                    }

                }

            }

        }

        if (matrix[0][0] == 0) {

        for (int j = 0; j <= col; j++) {

            matrix[0][j] = 0;

            }

        }

        if (col0 == 0) {

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

            matrix[i][0] = 0;

            }

        }

    }

}

Q8. Kth Smallest Element in a Sorted Matrix Given an n x n matrix where each of the rows and columns is sorted in ascending order, return the kth smallest element in the matrix. Note that it is the kth smallest element in the sorted order, not the kth distinct element. You must find a solution with a memory complexity better than O(n2).

class Solution {

    public int kthSmallest(int[][] matrix, int k) {

        int count=0;

        int[]arr=new int[matrix.length\*matrix[0].length];

        int l=0;

        int temp=0;

        for(int i=0;i<matrix.length;i++){

            for(int j=0;j<matrix[0].length;j++){

                arr[l++]=matrix[i][j];

            }

        }

        Arrays.sort(arr);

        return arr[k-1];

    }

}