1. Spiral Matrix

       class Solution {

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

        int left = 0;

        int right = matrix[0].length;

        int top = 0;

        int bottom = matrix.length;

        List<Integer> res = new ArrayList<Integer>();

        while(right>left && bottom>top){

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

                res.add(matrix[top][i]);

            }

            top++;

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

                res.add(matrix[i][right-1]);

            }

            right--;

            if((right>left && bottom>top)!=true){

                break;

            }

            for(int i=right-1;i>=left;i--){

                res.add(matrix[bottom-1][i]);

            }

            bottom--;

            for(int i=bottom-1;i>=top;i--){

                res.add(matrix[i][left]);

            }

            left++;

        }

        return res;

    }

}

1. K Weakest row in a matrix

class Solution {

    public int[] kWeakestRows(int[][] mat, int k) {

        PriorityQueue<int[]> pq= new PriorityQueue<>((a,b)->{

                if(a[1]==b[1])

                    return b[0]-a[0];

                return b[1] - a[1];

            });

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

            int s = 0;

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

                s+=mat[i][j];

            }

            if(pq.size()==k){

                if(pq.peek()[1]>s){

                    pq.poll();

                    pq.offer(new int[]{i,s});

                }

            }

            else{

                pq.offer(new int[]{i,s});

            }

        }

        int[]res = new int[k];

        for(int i=k-1;i>=0;i--)

            res[i] = pq.poll()[0];

        return res;

    }

}

1. Set Matrix Zero

class Solution {

    public void setZero(int [][] matrix,int row,int col){

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

            if(matrix[i][col]!=0)

                matrix[i][col]=0;

        }

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

            if(matrix[row][i]!=0)

                matrix[row][i]=0;

        }

    }

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

        List<int[]> zeroPositions = new ArrayList<>();

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

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

                if(matrix[i][j]==0)

                    zeroPositions.add(new int[]{i,j});

            }

        }

        for(int i=0;i<zeroPositions.size();i++){

            setZero(matrix,zeroPositions.get(i)[0],zeroPositions.get(i)[1]);

        }

    }

}

1. Matrix diagonal sum

class Solution {

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

        int sum=0;

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

            //primary diagonal

            sum += mat[i][i];

            //secondary diagonal

            if( i!= mat.length-1-i){

            sum += mat[i][mat.length-1-i];

            }

        }

        System.gc();

        return sum;

    }

}

1. Addition of two Square Matrices

class Solution

{

public void Addition(int[][] matrixA, int[][] matrixB)

{

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

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

matrixA[i][j]+=matrixB[i][j];

}

}

}

}

1. Multiply Two Matrices

public void Mutliply(int[][] matrixA, int[][] matrixB){

int rowsA = matrixA.length;

int colsA = matrixA[0].length;

int colsB = matrixB[0].length;

int[][] result = new int[rowsA][colsB];

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

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

for (int k = 0; k < colsA; k++) {

result[i][j] += matrixA[i][k] \* matrixB[k][j];

}

}

}

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

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

matrixA[i][j] = result[i][j];

}

}

}