SPIRAL MATRIX 2

class Solution {

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

int[][] ans = new int[n][n];

int top = 0, left = 0;

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

int k = 1;

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

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

ans[top][i] = k++ ;

}

top++;

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

ans[i][right] = k++;

}

right--;

if(top<=bottom){

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

ans[bottom][i] = k++;

}

bottom--;

}

if(left<=right){

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

ans[i][left] = k++;

}

left++;

}

}

return ans;

}

}

RICHEST CUSTOMER WEALTH

class Solution {

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

int max = Integer.MIN\_VALUE;

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

int sum = 0;

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

sum = sum + accounts[i][j];

}

max = Math.max(max, sum);

}

return max;

}

}

**Toeplitz Matrix**

class Solution {

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

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

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

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

return false;

}

}

}

return true;

}

}

MATRIX DIAGONAL SUM

class Solution {

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

int verticalSum=0;

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

verticalSum+=mat[i][i];

}

int count=0;

int horizontalSum=0;

for(int j=mat[0].length-1; j>=0; j--){

if(mat[0].length/2==count && mat[0].length%2!=0){

count++;

continue;

}

horizontalSum+=mat[count][j];

count++;

}

return (verticalSum+horizontalSum);

}

}

COUNT NEGATIVE NUMBERS IN SORTED MATRIX

class Solution {

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

int count = 0;

int m = grid.length;

int n = grid[0].length;

int row = 0;

int col = n - 1;

while (row < m && col >= 0)

{

if (grid[row][col] < 0)

{

count += m – row;

col--;

}

else

{

row++;

}

}

return count;

}

}

CONFIRMATION RATE

select s.user\_id, round(avg(if(c.action="confirmed",1,0)),2) as confirmation\_rate

from Signups as s left join Confirmations as c on s.user\_id= c.user\_id group by user\_id;

SET MATRIX ZEROES

public class Solution {

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

boolean fr = false,fc = false;

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

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

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

if(i == 0) fr = true;

if(j == 0) fc = true;

matrix[0][j] = 0;

matrix[i][0] = 0;

}

}

}

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

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

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

matrix[i][j] = 0;

}}

}

if(fr) {

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

matrix[0][j] = 0;

}

}

if(fc) {

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

matrix[i][0] = 0;

}

}

}}

K TH SMALLEST ELEMENT IN SORTED MATRIX

class Solution {

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

Map<Integer, Integer> map = new TreeMap<>();

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

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

map.put(matrix[i][j], map.getOrDefault(matrix[i][j], 0) + 1);

}

}

for(int n : map.keySet()) {

k -= map.get(n);

if(k <= 0) {

return n;

}

}

return -1;

}

}