**2-D Practice Questions**

**1-Spiral Traveral of a Matrix**

**Code:**

class Solution {

public List<Integer> spiralOrder(int[][] matrix) { ArrayList<Integer> result=new ArrayList<>(); int top=0; int left=0; int bottom=matrix.length-1; int right=matrix[0].length-1; while(top<=bottom && left<=right){ for(int i=left;i<=right;i++){ result.add(matrix[top][i]);

} top++; for(int i=top;i<=bottom;i++){ result.add(matrix[i][right]);

} right--;

if(top<=bottom){ for(int i=right;i>=left;i--){ result.add(matrix[bottom][i]);

} }

bottom--; if(left<=right){ for(int i=bottom;i>=top;i--){ result.add(matrix[i][left]);

} } left++; }

return result; // code here

}

}

**2-K weakest rows in a Matrix**

**Code:**

public int[] kWeakestRows(int[][] mat, int k) { int rows = mat.length;

int cols = mat[0].length;

int[] score = new int[rows];

int j;

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

j = 0;

for (; j < cols; j++) { if (mat[i][j] == 0) {

break;

}

}

/\*

* we can create a score to match the sort condition from description
* score = soldiersCount \* rows + currentRowIndex
* so we can get soldiersCount by score / rows, and get rowIndex by score % rows

\*/

score[i] = j \* rows + i;

}

Arrays.sort(score);

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

// get rowIndex

score[i] = score[i] % rows;

}

return Arrays.copyOfRange(score, 0, k); }

**3- Set Matrix Zeroes**

**Code:**

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;

}

}

}}

**4- Matrix Diagonal Sum**

**Code:**

class Solution {

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

int n = mat.length;

int ans = 0;

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

ans += mat[i][i];

ans += mat[n - 1 - i][i];

}

if (n % 2 != 0) {

ans -= mat[n / 2][n / 2];

}

return ans;

}

}

**5-Addition of two square matrices**

**Code:**

class Solution

{

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

{

int n = matrixA.length;

int m = matrixB[0].length;

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

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

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

}

}

}

}

**6-Multiply two matrices**

**Code:**

class Solution

{

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

{

int n=matrixA.length;

int res[][]=new int[n][n];

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

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

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

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

}

}

}

for(int i = 0; i < n; i++)

for(int j = 0; j < n; j++)

matrixA[i][j] = res[i][j];

}}