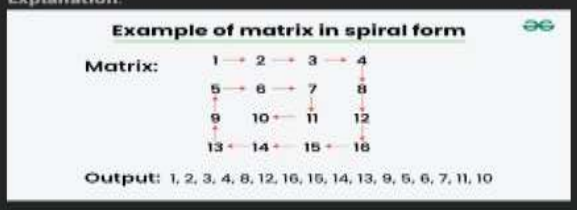
Matrix

1.Spirally traversing a Matrix

You are given a rectangular matrix **mat[][]** of size **n x m**, and your task is to return an array while **traversing** the matrix in **spiral** form.

Examples:

Input: mat[][] = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
Output: [1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]
Explanation:



Input: mat[][] = [[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12], [13, 14, 15, 16, 17, 18]]
Output: [1, 2, 3, 4, 5, 6, 12, 18, 17, 16, 15, 14, 13, 7, 8, 9, 10, 11]
Explanation: Applying same technique as shown above.

Input: mat[][] = [[32, 44, 27, 23], [54, 28, 50, 62]]
Output: [32, 44, 27, 23, 62, 50, 28, 54]
Explanation: Applying same technique as shown above, output will be [32, 44, 27, 23, 62, 50, 28, 54].

```java
// Function to return a list of integers denoting spiral traversal of matrix.
public ArrayList<Integer> spirallyTraverse(int matrix[][]) {
    // code here
    ArrayList<Integer> res=new ArrayList<>();
    int n=matrix.length;
    int m=matrix[0].length;

    int startRow=0;
    int endRow=n-1;
    int startCol=0;
    int endCol=m-1;

    while(startRow<=endRow && startCol<=endCol){
        for(int j=startCol;j<=endCol;j++){
            res.add(matrix[startRow][j]);
        }
        startRow +=1;
        for(int i=startRow;i<=endRow;i++){
            res.add(matrix[i][endCol]);

        }
        endCol -=1;

        if(startRow<=endRow){
            for(int j=endCol;j>=startCol;j--){
                res.add(matrix[endRow][j]);
            }
            endRow -=1;
        }
        //FIRST COL
        if(startCol<=endCol){
            for(int i=endRow;i>=startRow;i--){
                res.add(matrix[i][startCol]);
            }
            startCol +=1;
        }
    }
    return res;
```

2.Search a 2D-Matrix

You are given an `m x n` integer matrix `matrix` with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

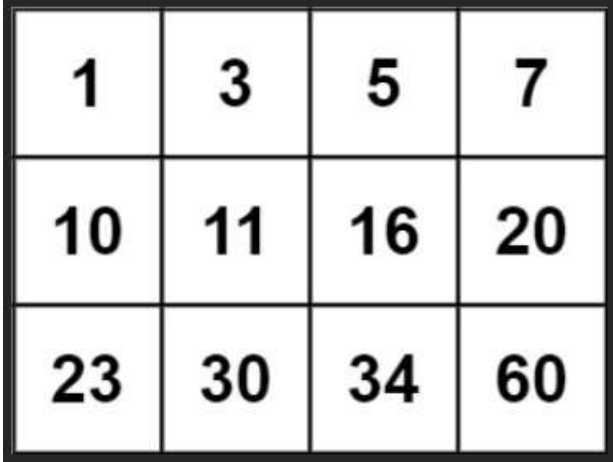
Given an integer `target`, return `true` *if* `target` *is in* `matrix` *or* `false` *otherwise.*

You must write a solution in `O(log(m * n))` time complexity.

Example 1:


```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true
```

Example 2:




```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13
Output: false
```

```java
public boolean searchMatrix(int[][] matrix, int target) {

    if(matrix==null || matrix.length==0 || matrix[0].length==0)
    {
        return false;

    }
    int m= matrix.length;
    int n= matrix[0].length;
    int start= 0;
    int end= m*n-1;
    while(start<=end)
    {
        int mid=(start+end)/2;
        int midX=mid/n;
        int midY=mid%n;
        if(matrix[midX][midY]==target)
        {
            return true;
        }
        if(matrix[midX][midY] < target)
        {
            start=mid+1;
        }
        else
        {
            end=mid-1;
        }
    }
    return false;
}
```

3.Median In a row-wise sorted matrix

Given a row-wise sorted matrix where the number of rows and columns is always **odd**, find the median of the matrix.

Examples:

Input: mat = [[1, 3, 5], [2, 6, 9], [3, 6, 9]]
Output: 5
Explanation: Sorting matrix elements gives us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.

Input: mat = [[1], [2], [3]]
Output: 2
Explanation: Sorting matrix elements gives us {1,2,3}. Hence, 2 is median

Input: mat = [[3], [5], [8]]
Output: 5
Explanation: Sorting matrix elements gives us {3,5,8}. Hence, 5 is median.

```java
class Solution {
    int median(int mat[][]) {
        // code here
        int n=mat.length;
        int m=mat[0].length;
        int res[]=new int[n*m];
        int index=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                res[index++]=mat[i][j];
            }
        }
        Arrays.sort(res);
        int ans=res[(n*m)/2];
        return ans;
    }
}
```

4.Sorted Matrix

Given an NxN matrix Mat. Sort all elements of the matrix.

Example 1:

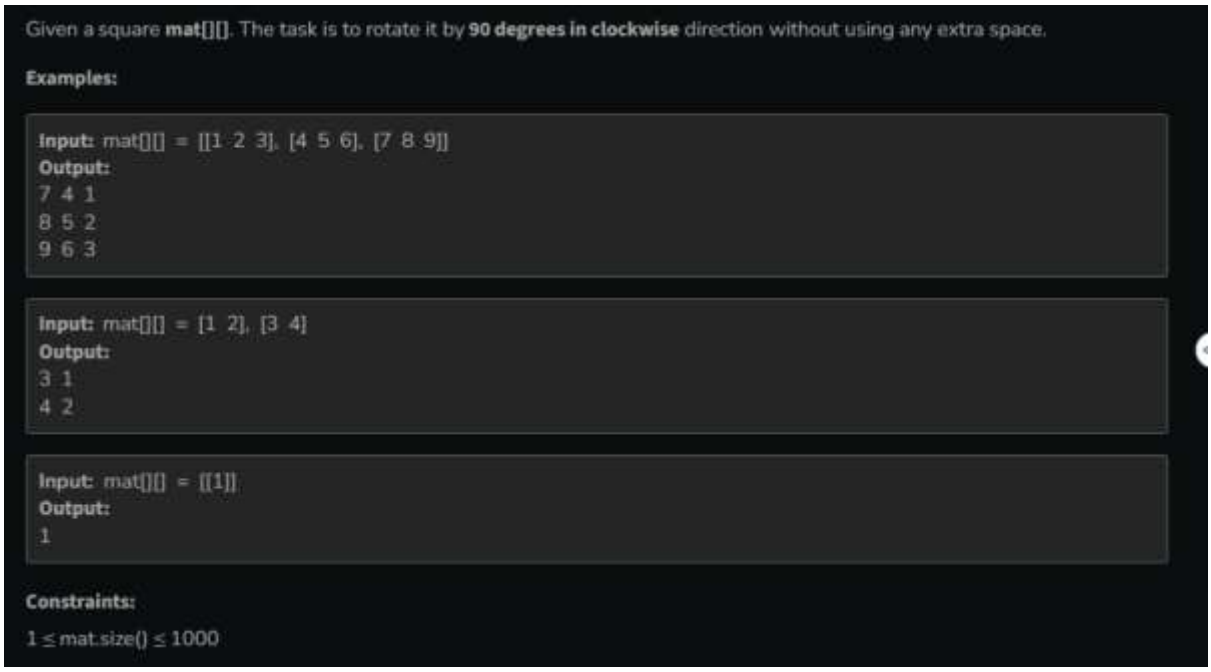
Input:
N=4
Mat=[[10,20,30,40],
[15,25,35,45]
[27,29,37,48]
[32,33,39,50]]
Output:
10 15 20 25
27 29 30 32
33 35 37 39
40 45 48 50
Explanation:
Sorting the matrix gives this result.

Example 2:

Input:
N=3
Mat=[[1,5,3],[2,8,7],[4,6,9]]
Output:
1 2 3
4 5 6
7 8 9
Explanation:
Sorting the matrix gives this result.

```
class Solution {
    int[][] sortedMatrix(int N, int Mat[][]) {
        // code here
        int arr[]=new int[Mat.length*Mat.length];
        int k=0;
        for(int i=0;i<Mat.length;i++){
            for(int j=0;j<Mat[0].length;j++){
                arr[k++]=Mat[i][j];
            }
        }
        Arrays.sort(arr);
        int l=0;
        for(int i=0;i<Mat.length;i++){
            for(int j=0;j<Mat[0].length;j++){
                Mat[i][j]=arr[l++];
            }
        }
        return Mat;
    }
};
```

5.Rotate By 90 degree(clockwise)



Given a square **mat[][]**. The task is to rotate it by **90 degrees in clockwise** direction without using any extra space.

Examples:

Input: mat[][] = [[1 2 3], [4 5 6], [7 8 9]]
Output:
7 4 1
8 5 2
9 6 3

Input: mat[][] = [1 2], [3 4]
Output:
3 1
4 2

Input: mat[][] = [[1]]
Output:
1

Constraints:

$1 \leq mat.size() \leq 1000$

```java
class GFG {
    static void rotate(int mat[][]) {
        // Code Here
          int n =mat.length;
        // reversing array
        for(int i=0;i<n/2;i++){ //need to swap only 1st and 3rd colum;
         int temp[]= mat[i];
         mat[i]= mat[n-1-i];
         mat[n-i-1]=temp;
        }
        //transposing matrix lets goo
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                int temp = mat[i][j];
                mat[i][j]= mat[j][i];
                mat[j][i]=temp;
            }
        }
    }
}
```

If I want to find for 180,270 degree clockwise just need to call two time 90 degree function in 180 degree function

And for 270 degree call 3 times 90 degree clockwise function.

Tricks -→ 90 Degree (Anti-clockwise)

```
public static  void rotate90Anticlockwise(int matrix[][]){
    int n=matrix.length;
    for(int i=0;i<n/2;i++){
        for(int j=i;j<n-i-1;j++){
            int temp=matrix[i][j];
            matrix[i][j] = matrix[j][n - i - 1];
            matrix[j][n - i - 1] = matrix[n - i - 1][n - j - 1];
            matrix[n - i - 1][n - j - 1] = matrix[n - j - 1][i];
            matrix[n - j - 1][i] = temp;
        }
    }

}
```

Same here also

For 180 degree Anticlockwise→ call 2 time 90 degree

For 270 degree Anticlockwise → call 3 times 90 degree

6.Max Rectangle

Given a binary matrix **mat[][]** of size **n * m**. Find the maximum area of a rectangle formed only of **1s** in the given matrix.

Examples:

Input: mat[][] = [[0, 1, 1, 0],
[1, 1, 1, 1],
[1, 1, 1, 1],
[1, 1, 0, 0]]
Output: 8
Explanation: The largest rectangle with only 1's is from (1, 0) to (2, 3) which is
[1, 1, 1, 1]
[1, 1, 1, 1]
and area is 4 *2 = 8.

Input: mat[][] = [[0, 1, 1],
[1, 1, 1],
[0, 1, 1]]
Output: 6
Explanation: The largest rectangle with only 1's is from (0, 1) to (2, 2) which is
[1, 1]
[1, 1]
[1, 1]

```java
/*Complete the function given below*/
class Solution {

    static int maxArea(int mat[][]) {

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

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

                if(mat[i][j] != 0){
                    mat[i][j] += mat[i-1][j];
                }
            }
        }

        int maxArea = 0;
        for(int i = 0 ; i < mat.length ; i++){
            maxArea = Math.max(maxArea, CurrMax(mat[i]));
        }

        return maxArea;
    }

    static int CurrMax(int[] arr){

        int n = arr.length;
        int[] nse = new int[n];
        Stack<Integer> st1 = new Stack<>();
```

```
53          int n = arr.length;
54          int[] nse = new int[n];
55          Stack<Integer> st1 = new Stack<>();
56
57          for(int i = n-1 ; i >= 0 ; i--){
58              while(st1.size() > 0 && arr[i] <= arr[st1.peek()]){
59                  st1.pop();
60              }
61              if(st1.size() == 0){
62                  nse[i] = n;
63              }
64              else{
65                  nse[i] = st1.peek();
66              }
67              st1.push(i);
68          }
69
70          int[] pse = new int[n];
71          Stack<Integer> st2 = new Stack<>();
72
73          for(int i = 0 ; i < n ; i++){
74              while(st2.size() > 0 && arr[i] <= arr[st2.peek()]){
75                  st2.pop();
76              }
77              if(st2.size() == 0){
78                  pse[i] = -1;
79              }
80              else{
81                  pse[i] = st2.peek();
82              }
83              st2.push(i);
84          }
85
86          int max = 0;
87          for(int i = 0 ; i < arr.length ; i++){
88              int width = nse[i] - pse[i] - 1;
89              int area = arr[i] * width;
90              max = Math.max(area, max);
91          }
92
93          return max;
```

Custom Input

7.Maximum Diffence between pair in a matrix

Given an **n x n** matrix, **mat[n][n]** of integers. The task is to find the maximum value of **mat(c, d)- mat(a, b)** over all choices of indexes such that both **c > a** and **d > b**.

Example 1:

Input: mat[N][N] = {{ 1, 2, -1, -4, -20 },
{ **-8**, -3, 4, 2, 1 },
{ 3, 8, 6, 1, 3 },
{ -4, -1, 1, 7, -6 },
{ 0, -4, **10**, -5, 1 }};
Output: 18
Explanation: The maximum value is 18 as mat[4][2] - mat[1][0] = 18 has maximum difference.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **findMaxValue()** which takes a matrix mat and returns an integer as output.

```java
class Solution {
    public static int findMaxValue(int n, int[][] mat) {
        // code here

        int m = mat[0].length;

        // Preprocessing first column
        for (int i = 1; i < n; i++) {
            mat[i][0] = Math.min(mat[i][0], mat[i - 1][0]);
        }

        // Preprocessing first row
        for (int j = 1; j < m; j++) {
            mat[0][j] = Math.min(mat[0][j], mat[0][j - 1]);
        }

        int ans = Integer.MIN_VALUE;

        // Main logic
        for (int i = 1; i < n; i++) {
            for (int j = 1; j < m; j++) {
                int h = mat[i][j] - mat[i - 1][j - 1];
                ans = Math.max(ans, h);
                int a = mat[i][j];
                mat[i][j] = Math.min(mat[i - 1][j], Math.min(mat[i][j - 1], mat[i - 1][j - 1]));
                mat[i][j] = Math.min(mat[i][j], a);
            }
        }

        return ans;

    }
}
```

8.Kth Element in matrix

Given a N x N matrix, where every row and column is sorted in non-decreasing order. Find the kth smallest element in the matrix.

Example 1:

Input:
N = 4
mat[][] = {{16, 28, 60, 64},
{22, 41, 63, 91},
{27, 50, 87, 93},
{36, 78, 87, 94 }}
K = 3
Output: 27
Explanation: 27 is the 3rd smallest element.

Example 2:

Input:
N = 4
mat[][] = {{10, 20, 30, 40}
{15, 25, 35, 45}
{24, 29, 37, 48}
{32, 33, 39, 50}}
K = 7
Output: 30
Explanation: 30 is the 7th smallest element.

```
// User function Template for Java
class Solution
{
    private static int upperBound(int[] row, int value)
    {
        int low=0;
        int high=row.length;

        while(low<high)
        {
            int mid = low+(high-low)/2;
            if(row[mid]<=value)
            {
                low=mid+1;
            }
            else
            {
                high=mid;
            }
        }
        return low;
    }
    private static int countLessOrEqual(int[][] mat,int value)
    {
        int n=mat.length;
```

```
        return low;
    }
    private static int countLessOrEqual(int[][] mat,int value)
    {
        int n=mat.length;

        int count=0;
        for(int i=0;i<n;i++)
        {
            int lb=upperBound(mat[i],value);
            count+=lb;
        }
        return count;
    }

    public static int kthSmallest(int[][]mat,int n,int k)
    {
        //code here.
        int low = mat[0][0];
        int high = mat[n-1][n-1];

        while(low<high)
        {
            int mid = low+(high-low)/2;

            int count = countLessOrEqual(mat,mid);

            if(count<k)
            {
                low=mid+1;
            }
            else
            {
                high=mid;
            }
        }
        return low;
    }
}
```

9.Common Elements in all rows of a given matrix

Given an m x n matrix, find all common elements present in all rows in O(mn) time and one traversal of matrix.

Example:

Input:

```
mat[4][5] = {{1, 2, 1, 4, 8},
             {3, 7, 8, 5, 1},
             {8, 7, 7, 3, 1},
             {8, 1, 2, 7, 9},
            };
```

Output:

```
1 8 or 8 1
8 and 1 are present in all rows.
```

```java
import java.util.*;

class GFG
{

// Specify number of rows and columns
static int M = 4;
static int N =5;

// prints common element in all rows of matrix
static void printCommonElements(int mat[][])
{

    Map<Integer,Integer> mp = new HashMap<>();

    // initialize 1st row elements with value 1
    for (int j = 0; j < N; j++)
        mp.put(mat[0][j],1);

    // traverse the matrix
    for (int i = 1; i < M; i++)
    {
        for (int j = 0; j < N; j++)
        {
            // If element is present in the map and
            // is not duplicated in current row.
            if (mp.get(mat[i][j]) != null && mp.get(mat[i][j]) == i)
            {
                // we increment count of the element
                // in map by 1
                mp.put(mat[i][j], i + 1);

                // If this is last row
                if (i == M - 1)
                    System.out.print(mat[i][j] + " ");
            }
        }
    }
}

// Driver code
public static void main(String[] args)
```