MEDIAN IN A ROW-WISE SORTED 2D MATRIX

☆ In this problem, we are given a 2D matrix where each row is sorted individually and our job is to return the row, col of the median which would exist if the 2D matrix was flatenned into a 1D array and was sorted.

Brute force solution is to run an M×N iteration, add each element to a 1D array, sort the 1D array and then return its median. Time complexity is $O(MN + MN \log MN)$

Optimal Solution involves the use of Binary Search. Our answer will always be between the minimum and maximum of the matrix. Our goal will be to figure out the first element which has exactly or atleast $\frac{MN}{2}$ numbers

<= itself.

Pseudocode :

```
median2DRowWise(arr, M, N) {
    low = findLowest(arr, M, N)
    high = findHighest(arr, M, N)
    while(low <= high) {
```

```
        mid = (low + high) / 2
        smallerEquals = uB(arr, M, N, mid)
        if (smallerEquals <= MN/2) {
            low = mid + 1
        } else {
            high = mid + 1
        }
    }
    return low
}
uB(arr, M, N, K) {
    ans = 0
    for (i = 0 → M - 1) {
        low = 0
        high = N - 1
        uB = 0
        while (low <= high) {
            mid = (low + high) / 2
            if (arr[i][mid] <= K) {
                uB = K
                low = mid + 1
            } else {
                high = mid - 1
            }
        }
        ans += uB
    }
    return ans
}
findLowest(arr, M, N) {
    low = INT_MAX
    for (i = 0 → M - 1) {
        if (arr[i][0] < low) {
```

```
            low = arr[i][0]
        }
    }
    return low
}
findHighest(arr, M, N) {
    high = INT_MAX
    for (i = 0 -> M-1) {
        if (arr[i][N-1] > high) {
            high = arr[i][N-1]
        }
    }
    return high
}
```