MEDIAN OF ROW WISE SORTED MATRIX

given a row-wise matrix M x N where m x n is odd
. find the median in the given matrix.

1	4	9
2	5	6
3	8	7

Result (5)

as 1 2 3 4 (5) 6 7 8 9
↳ mid

BRUTE

→ put element in 1D array
→ then sort the array and find mid element.

TC → O(n², nlogn) O(n×m + log(n×m)(n×m))
↳ traverse ↳ sort

SC → ~~O(n²)~~ O(n×m)

OPTIMAL

$$\begin{bmatrix} 1 & 5 & 7 & 9 & 11 \\ 2 & 3 & 4 & 5 & 10 \\ 9 & 10 & 12 & 14 & 16 \end{bmatrix}$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	2	3	4	(6)	6	7	7	9	11	12	13	13	14	14	15

↳ how many number ≤ to that no in the sorted array.

{1, 2, 3, 4, 5, 5} = 6

now median = $\frac{n \times m}{2} = \frac{4+3}{2} \frac{5 \times 3}{2} = \frac{15}{7} = (7)$

∴ there will be seven elements before and after median

even if ↑ have all elements are Median

9 9 9 9 9 9 9 (9) 10 11 12 13 14 15 16

we can surely say that median would be that element whose [no of elements ≤ that element] > 7

1 2 3 4 5 6 7 8 9 11 1

∴ in previous example (9) → whose no of element ≤ 9 is (9) which > 7 will be our answer

∴ we will find the first occurence of element whose [no of element ≤ that element] > 7 ⤳ $\frac{m \times n}{2}$

```
f( ){
  low = ⤳ go through the first and last colum to get the lowest and highest
  high =
  req = (m x n)/2
  while (low <= high) {
      mid = (low + high) / 2;
      smallerEquals = ⊕ compute(mat[], mid);
      if (smallerEquals < req) low = mid+1;
      else high = mid - 1 ;}
  }
return low;
```

compute [] {

$$\begin{bmatrix} 1 & 5 & 7 & 9 & 11 \\ 2 & 3 & 4 & 5 & 10 \\ 9 & 10 & 12 & 14 & 16 \end{bmatrix}$$

0 → find upperbound(mat[0], 8) = 3 (indx) ∴ 3 elem ≤ 8
(8 → n)

1 → upper(mat[1], 8) = 4 ∴ 4

2 → " (mat[2], 8) = 0 ∴ 0

∴ 3 + 4 + 0 = 7 elements

∴ {

```
count = 0;
for (i = 0 → n)
    count += upperbound(mat[i], n);
        ↳ binary search
}
return count;
}
```

$O(n \times \log_2 m)$

compute function

TC → $O(\log(10^9) \times n \times \log_2 m)$

(10^9 → max search space constraints)