Interview question

Search 2D Matrix

↳ Row-wise sorted from left to right

↳ first integer of each row > Last integer of the previous row

2D Matrix →

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

3x4

m = 3, n = 4

m → # rows (no. of rows)

n → # columns (no. of columns)

target = 60

left = 0 (starting index)

right = m x n - 1

= 3 x 4 - 1

= 11 (ending index)

target = 3 → true

target = 22 → false

Brute force Approach →

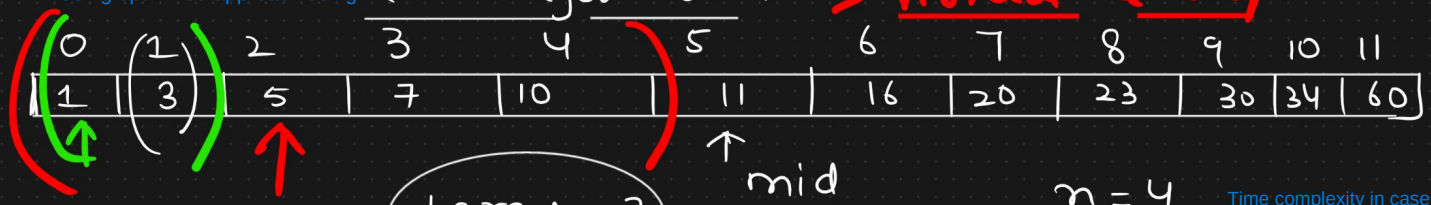
O(m * n)

```
for i = 0 to m-1:          (m times)
    for j = 0 to n-1:      (n times)
        if arr[i][j] == target:
            return True
return false
```

Third iteration

mid = (0 + 1)/2 = 0

#Using optimized approach using Row-major form → Virtual array



target = 3

Second iteration

(0+4)//2 = 2

row = 2//4 = 0

col = 2%4 = 2

n = 4

Time complexity in case on using Binary Search approach

O(log mn)

(Binary search)

First iteration using Binary Search approach

mid = 0 + (11 - 0)//2 = 5

Index of middle element when 2D arranged in linear or 1D fashion

row ← 5//4 = 1

row index of mid element in 2D array= (mid element//no. of columns)

column ← 5%4 = 1

column index of mid element in 2D array= (mid element%no. of columns)

left = 0

right = mid - 1

If element is not found then change the start and end pointers

May be if we are arranging in Column major form then for getting row and column of indexes of mid value we would require to take division and modulo with no. of rows for generating the respective row and col index for middle element

Please note that here in Virtual array we are basically producing the Row major form array by arranging 2 D in linear fashion using there respective array indices.

First thing that should come in your mind when introduced with searching problem statement is to look for presence of sorted array. If present then simply try applying the Binary Search algorithm on top of it.