BINARY SEARCH IN 2D ARRAYS

Searching in Matrices -

	0	1	2
0	18	9	12
1	96	-4	(91)
2	44	33	16

Return the index of target element
target = 91 (Array is not Sorted)

Keep checking one by one
Run a nested for loop

```
int [] ans = new int [2];
for (int row = 0; row < n ; row++) {
    for (int col = 0 ; col < n ; col++){
        if ( arr [row] [col] == target) {
            ans [0] = row ;
            ans [1] = col ;
            return ans ; // found ans  }
    }}
    return -1 ;  // After running the loop , not found the answer
```

Worst number of steps it's going to take - N * N = N^2 = $O(N^2)$
when, rows = n & cols = m the complexity will be O (N * M)

Q: Matrix is sorted in a row wise & colcum wise manner.
Return true if found the target. target = 37

Lower bound →	0	1	2	3 ← [Upper Bound]
0	10	20	30	40
1	15	25	35	45
2	28	29	(37)	49
3	33	34	38	50

In matrix related Questions when you are trying to reduce the search space
↓
Think about how you can eliminate rows & cols
We are going to compare with - Upper Bound

① ☑ Start checking from the last column.
☑ Let's compare, Binary Search
① 40 > 37 (i.e. target) ✓
② 40 < 37
③ 40 == 37

40 ← upper bound
45 ← All the elements
49 that you have
50 in the column

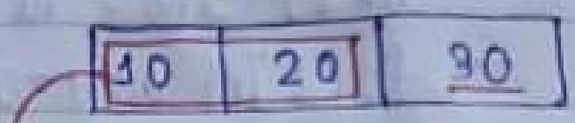
→ greater than 37 ←

☑ All elements in the col > 40 > target
☑ So, the target element will not be there Because all the elements are greater than it. So, we will ignore / eliminate the column
↓
It will reduce the search space

②

	0	1	2	3
0	10	20	30	40
1	15	25	35	45
2	28	29	37	49
3	33	34	38	50

- Now we will consider col = 2 as last column
- Here, 30 is upperbound. Let's compare
  - ① 30 > 37
  - ② 30 < 37 ✓
  - ③ 30 == 37



- All the numbers in the row < 30 < 37
- So, the target will not be there as all the numbers are less than target. So, we will ignore the Row

③

- Now, col = 2 is the last column &
  Row = 1 is the first row (consider)
- Upperbound = 35. Let's compare.
  - ① 35 == 37
  - ② 35 > 37
  - ③ 35 < 37 ✓ ← lessthan means eleminate row

↓ It will become

0	~~10~~	~~20~~	~~30~~	~~40~~
1	15	25	35	~~45~~
2	28	29	37	~~49~~
3	33	34	38	~~50~~

lessthan37 →

15	25	35

- All the members in the row < 35 < 37
- Eleminate the row.

④

	0	1	↓ 2	3
0	~~10~~	~~20~~	~~30~~	~~40~~
1	~~15~~	~~25~~	~~35~~	~~45~~
2	28	29	37	~~49~~
3	33	34	38	~~50~~

- Now, last column = 2, first row = 2
- Upperbound = 37, let's compare.
  - ① 37 == 37 ✓ ← (Found the Target)
  - ② 37 > 37
  - ③ 37 < 37

* Case 1 → If element == target → Are Found
  Case 2 → if element > target → eleminate column → col--
  Case 3 → if element < target → eliminate row → row++
* We are going to search from row zero and column last
* Keep running the loop till the row is actually less than the length & column is actually greater than equal to zero
* It's making comparisons n+n = 2n times
  time complexity = O(N) (because constants are removed in complexity)
  Space complexity = O(1)
  (we are not taking any extra space)

Program →

```
public class RowColMatrix {
    public static void main (String [] args) {
        int [][] arr = {
                { 10, 20, 30, 40 },
                { 15, 25, 35, 45 },
                { 28, 29, 37, 49 },
                { 33, 34, 38, 50 }
            };
System.out.println( Arrays.toString (search (arr, 49 )));
    }

static int[] search (int [][] matrix , int target ) {
    int row = 0 ;
    int col = matrix [0].length -1;
while ( row < matrix.length && col >= 0) {
    if ( matrix [row] [col] == target) {
        return new int[] { row, col } ;
        }
    if( matrix [row] [col] < target) {
        row++ ;
        }
    else { col-- ; }
    }
return new int [] {-1, -1 } ;
    }
}
```