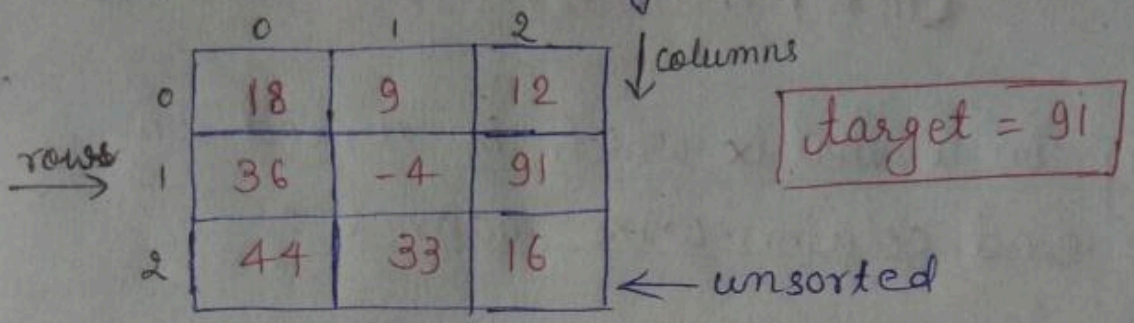
Binary Search in 2D Arrays

* Searching in Matrices :-

Lets take a 2D Array (or Matrix)



target = 91

NOTE: We have already learned "HOW TO TRAVERSE IN 2D ARRAYS".

>>> Initially: row = 0, col = 0
check: Is, 18 = 91 ⇒ NO ⇒ col++.

Now, row = 0, col = 1
check: Is, 9 = 91 ⇒ No ⇒ col++

Now, row = 0, col = 2
check: Is, 12 = 91 ⇒ No ⇒ loop over
⇒ row++

Now, row = 1, col = 0
check: Is, 36 = 91 ⇒ No ⇒ col++

row = 1, col = 1
check: Is, -4 = 91 ⇒ No ⇒ col++

row = 1, col = 2
check: Is, 91 = 91 ⇒ Yes ⇒ Ans found

So, Ans = [1, 2]
↑ row ↑ column

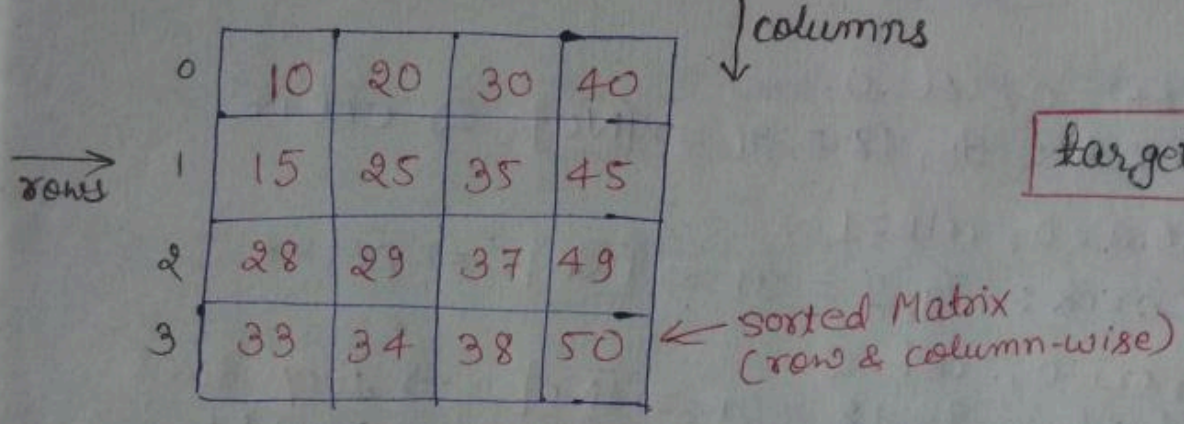
* Worst case Time Complexity

$O(N^2)$ → if the matrix is of size N×N

$O(N \times M)$ → if the matrix is of size N×M
(Here, N = no of rows, M = no of columns)

* Searching in a matrix which is sorted in row-wise and column-wise manner :—

Let's take a matrix in which every row and every column is sorted :-



target = 37

>>> There are 3 cases

case 1 : if element == target
⇒ ans found

case 2 : if element < target
⇒ row++

case 3 : if element > target
⇒ col --

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

lower bound → (0,0) = 10

Upper bound (this last column) → 40

target = 37

>> Start checking from last column :-

- Since, 40 > 37 ⇒ all the elements in this last column is greater than target element (i.e, 37) Because, the matrix is sorted row & column-wise.
  ⇒ So, col-- (ignore the last column)

⇒ Now, our Matrix is —

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

LB → (0,0) = 10

>>> checking in 0th row:

- Since, 30 < 37 ⇒ Means, all the left-hand side elements of 30 in row 0 is smaller than 30 (as matrix is sorted row-wise), So, obviously that elements are smaller than target element (i.e 37).
  ⇒ So, row++ (ignore 0th row)

⇒ Now, our matrix is :

15	25	35
28	29	37
33	34	38

>>> Checking in 1st row :-

- Since, `35 < 37` ⟹ All the left-side elements of 35 is smaller than 35 (As matrix is sorted) means → That elements are also smaller than target element (i.e, 37)

  ⟹ So, `row++` (Ignore 1st row)

⟶ Now, our matrix becomes:

28	29	37
33	34	38

>>> checking in 2nd row :-

Since, `37 == target` ⟹ Ans found at [2,2]

* NOTE : We keep running the loop till row is less than length of matrix and column is greater, equal to zero(0).

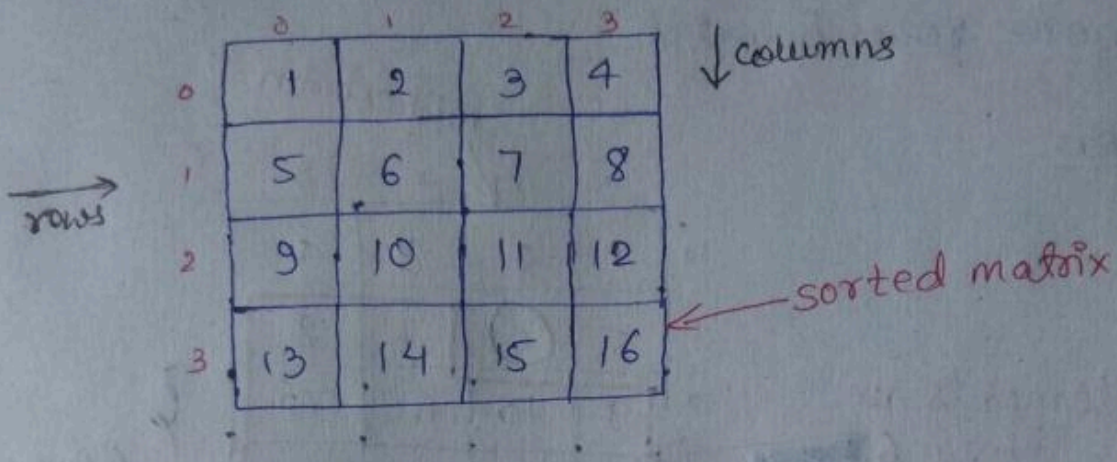
* Time Complexity = O(N)

* Space Complexity = O(1) //constant

Remember : In these type of Matrices problem, try to reduce the search space. i.e, eleminating the rows and columns.

* Searching in a Sorted Matrix :—

Let's take a sorted matrix —



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

*** Approach**

Take middle column and perform binary search on it
Then we have 3 cases

- There are 3 cases :—

case 1 : If element == target
// ans

case 2 : If element > target
// ignore rows after it (below rows)

case 3 : If element < target
// ignore above rows

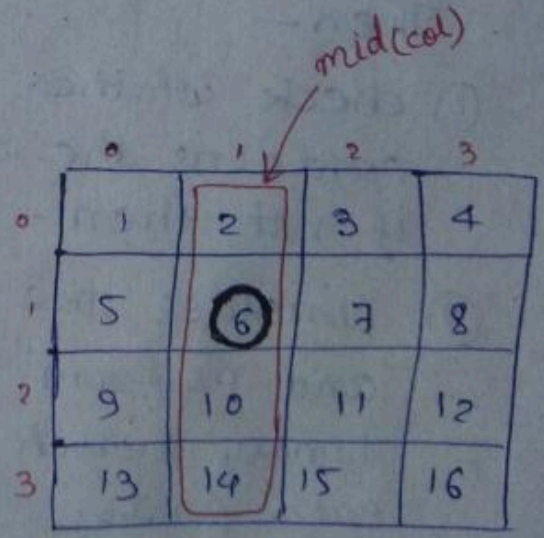
>>> Let's take **target = 3**

mid(col) = $\frac{0+3}{2}$ = 1

Now, find mid of that column 1 (i.e, Perform binary search on mid column)

mid(col1) = $\frac{0+3}{2}$ = 1

i.e, element 6



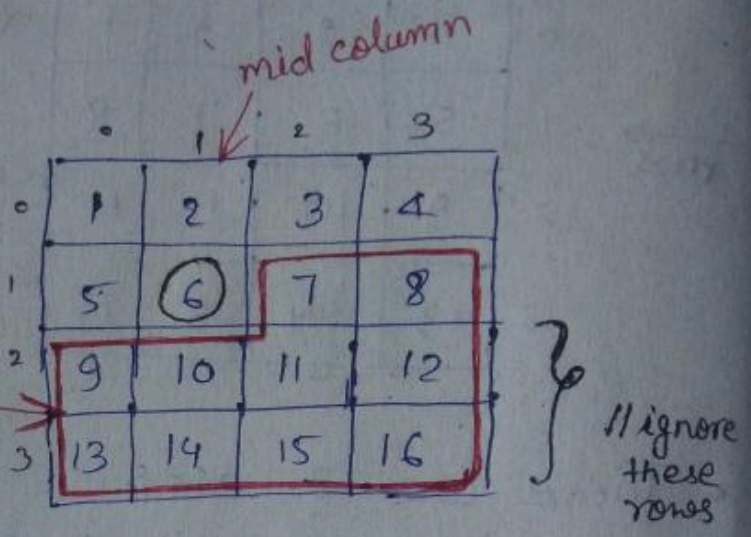
	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

Now, check the above mentioned cases —

Here, $\boxed{6 > 3}$ i.e case 2 : element > target

⇒ So, ignore rows below it

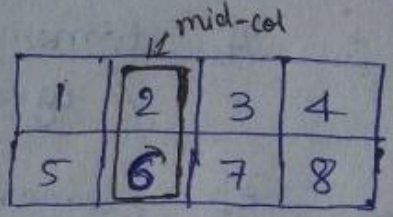
Explaination



We, know
These elements are greater than 6
(becoz, the matrix is sorted)

So, obviously all these nos. are also greater than target (i.e 3). So, ignore below rows.

Now, Matrix becomes →

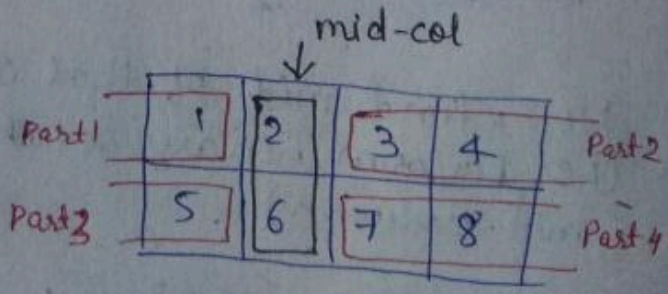


case

>>> In the end, 2 rows are remaining
Then —

① check whether the mid column you are at contains the answer
if not then —

② consider the 4 parts and perform simple binary search in each parts.



* Time Complexity = $O(\log(N) + \log(M))$

$\log(N)$ ↑ search across rows

$\log(M)$ ↑ search across columns