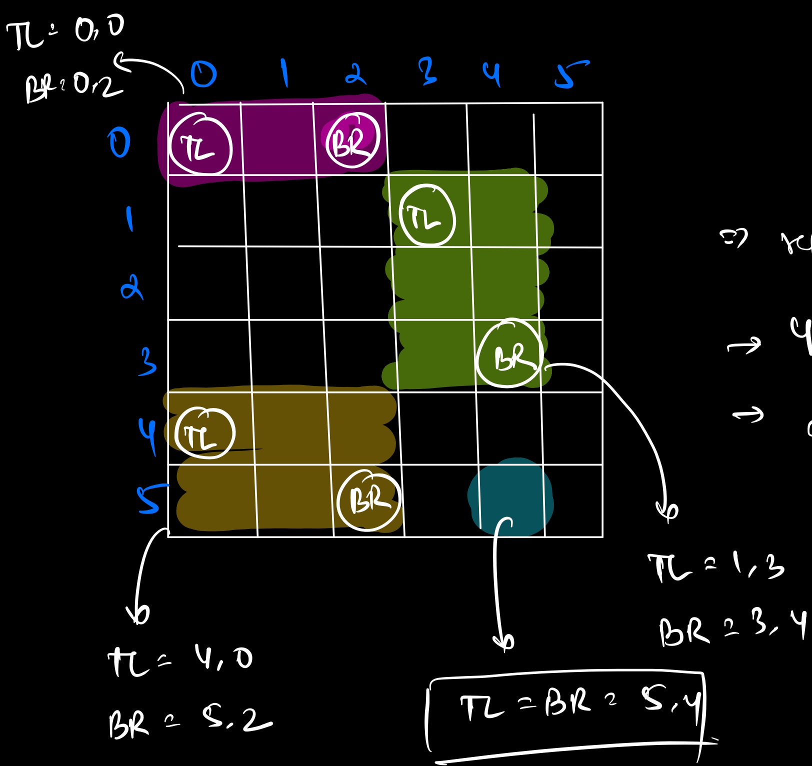
Q1. Given a matrix of N x M and Q queries. Find the submatrix sum of every query. [TL, BR]

TL → top left

BR → bottom right



⇒ represent a sub matrix

→ 4 corners

→ 2 corner (TL, BR)

Bruteforce

i) take each query

ii) traverse entire sub matrix for each query

$$\boxed{\begin{array}{l} TC \Rightarrow O(Q \times N \times M) \\ SC \Rightarrow O(1) \end{array}}$$

Optimised

array - 1D - queries (sum l to R)

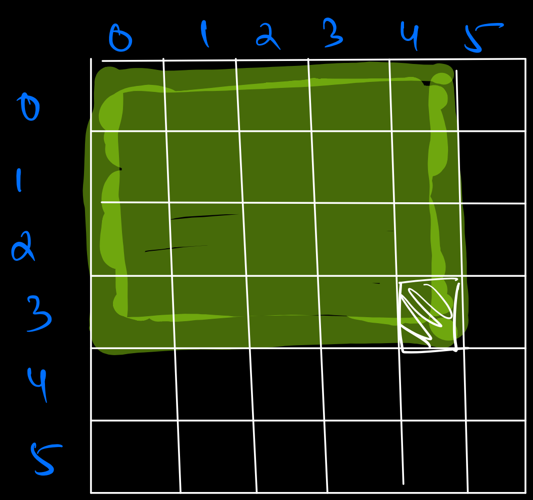
pfsum

arr[i] ⇒ pfsum[i] ⇒ sum of elements from 0 to i

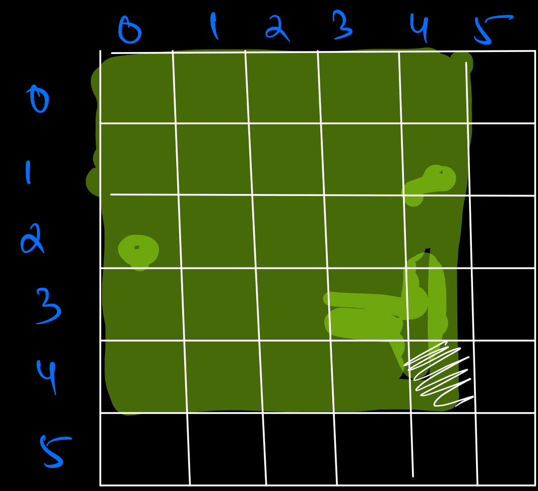
1D

arr[i][j] ⇒ pfsum[i][j] ⇒ sum of all elements from (0,0) to (i,j)

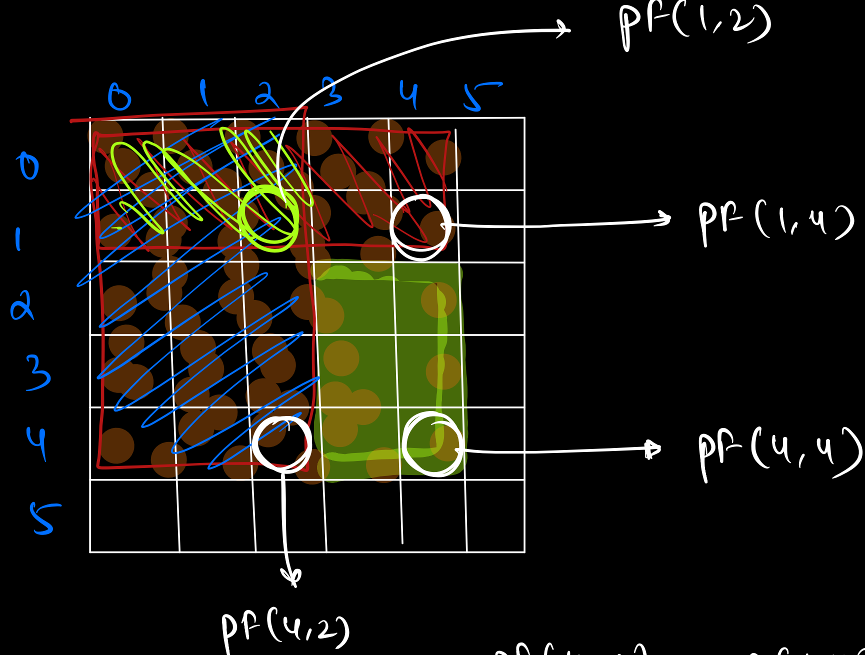
2D



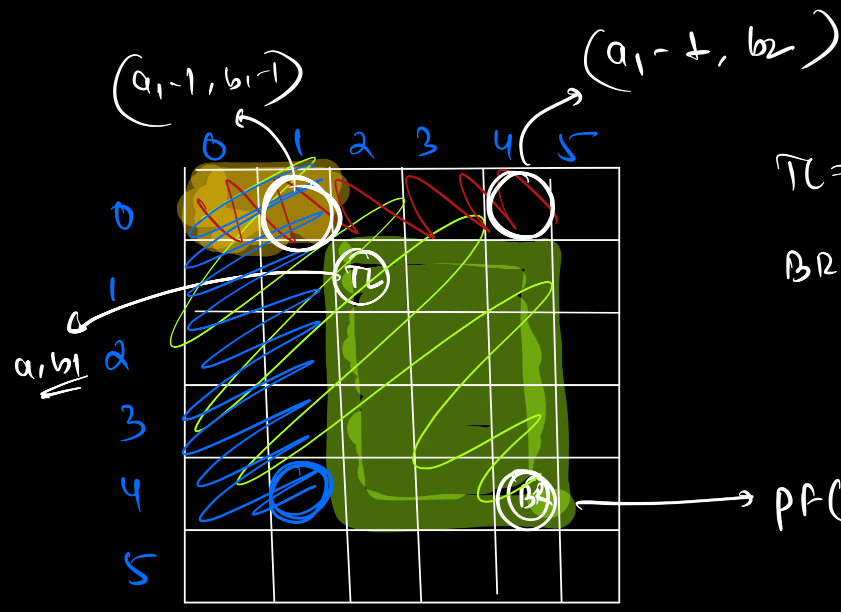
pfsum[3,4]



pfsum[4,4]



Sum of submatrix [2,3] to [4,4] $= pf(4,4) - pf(1,4) - pf(4,2) + pf(1,2)$



$pf(a_2, b_2)$

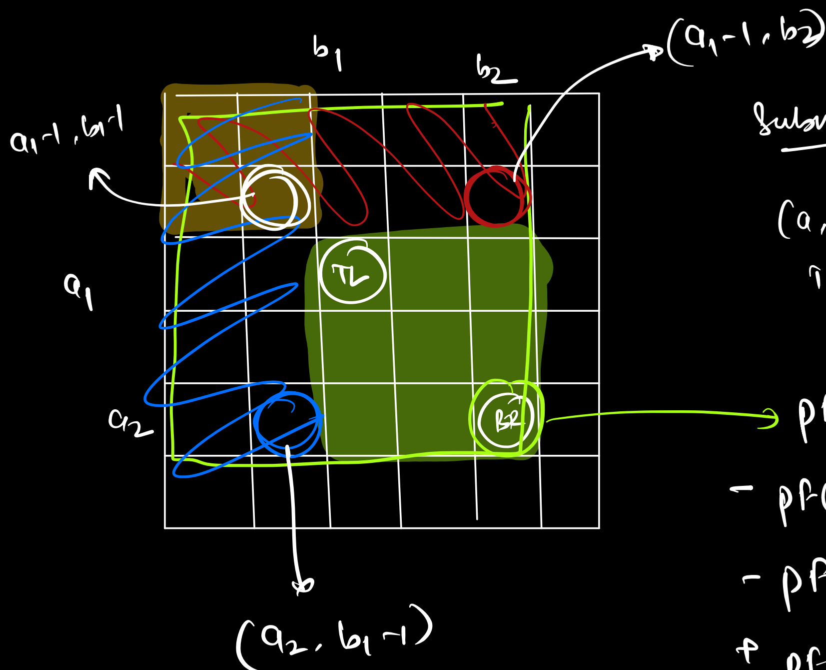
$TL = a_1, b_1$

$BR = a_2, b_2$

Sum of submatrix (a_1, b_1) to (a_2, b_2) $= pfSum(a_2, b_2) - pfSum(a_1-1, b_2)$

$- pfSum(a_2, b_1-1)$

$+ pfSum(a_1-1, b_1-1)$



Submatrix

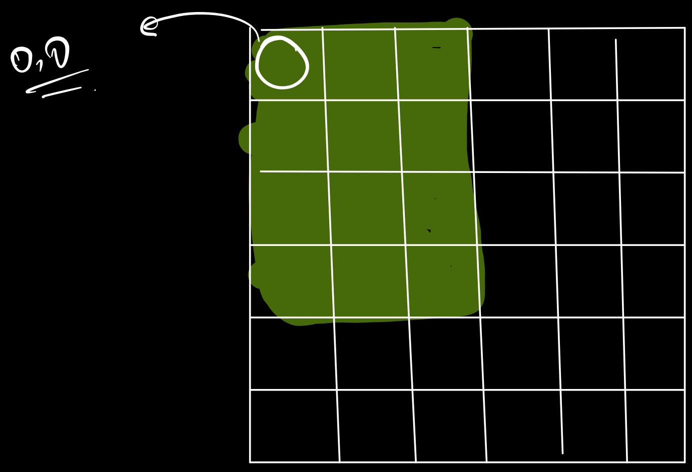
(a_1, b_1) TL (a_2, b_2)

$pfSum(a_2, b_2)$

$- pf(a_1-1, b_2)$

$- pf(a_2, b_1-1)$

$+ pf(a_1-1, b_1-1)$



sum = pf(a_2, b_2)

if (a_1 > 0)

  sum = sum − pf(a_1 − 1, b_2)

if (b_1 > 0)

  sum = sum − pf(a_2, b_1 − 1)

if (a_1 > 0 && b_1 > 0)

  sum = sum + pf(a_1 − 1, b_1 − 1)

O(1)

TC ⇒ finding PFsum → O(N×M)

then iterating through all queries

O(Q)

$\boxed{SC \Rightarrow O(N \times M)}$ $\boxed{TC \Rightarrow O(NM + Q)}$

* creating a prsum matrix

a_0	b_0	c_0
a_1	b_1	c_1
a_2	b_2	c_2

$\xrightarrow{\text{prsum rowise}}$

a_0	a_0+b_0	$a_0+b_0+c_0$
a_1	a_1+b_1	$a_1+b_1+c_1$
a_2	a_2+b_2	$a_2+b_2+c_2$

$\downarrow$

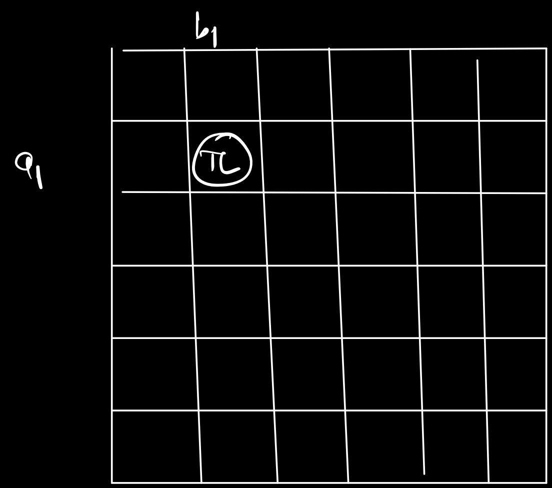
prsum column

a_0	a_0+b_0	$a_0+b_0+c_0$
a_0+a_1	$a_0+b_0+a_1+b_1$	$a_0+b_0+c_0+a_1+b_1+c_1$
$a_0+a_1+a_2$	$a_0+b_0+a_1+b_1+a_2+b_2$	$a_0+b_0+c_0+a_1+b_1+c_1+a_2+b_2+c_2$

TC to calculate prsum

$O(N \times M)$

Q2. Given a matrix $N \times M$. Find sum of all submatrix sums.



submatrix can be represented by TL, BR

all possible combo's of TL & BR = no. of submatrices

* all the cells can act as top left.

```
// top left
for(a1=0; a1<N; a1++){                 ] → N×M
    for( b1=0; b1<M; b1++) {
        // (a1, b1) top left

        for( a2=a1 ; a2<N; a2++) {     ] N×M
            for( b2=b1; b2<M; b2++){
                // (a2, b2) bottom right

                sum [(a1, b1) - (a2 b2)]
} }   }
```

1) find sum of all possible submatrices and add them

$O(N \times M \times N \times M)$

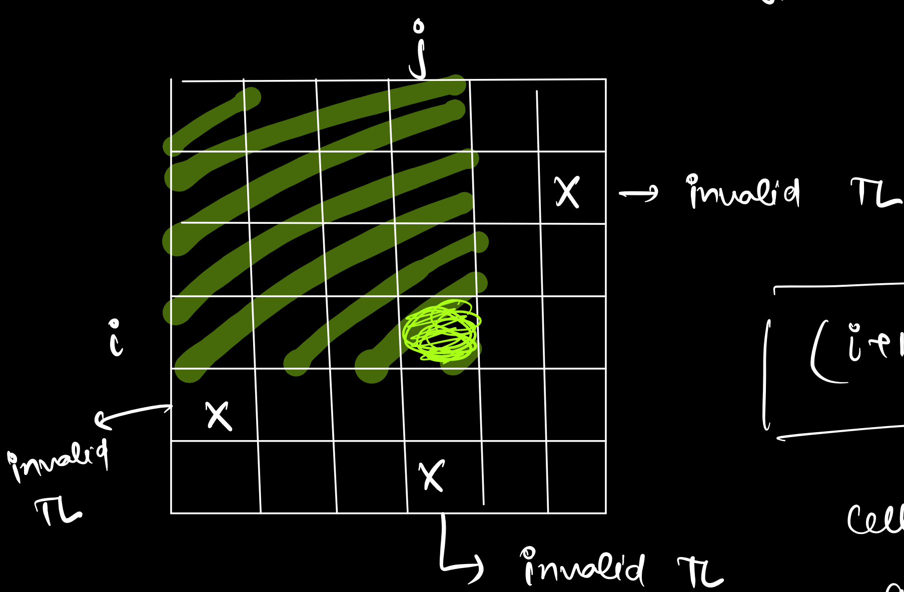
TC $\Rightarrow \boxed{O(N^2M^2)}$

$\Rightarrow$ contribution technique

Count $\Rightarrow$ In how many submatrices, a particular cell is present.

contribution by cell (i,j) $\Rightarrow$ # count $\times$ arr[i][j]

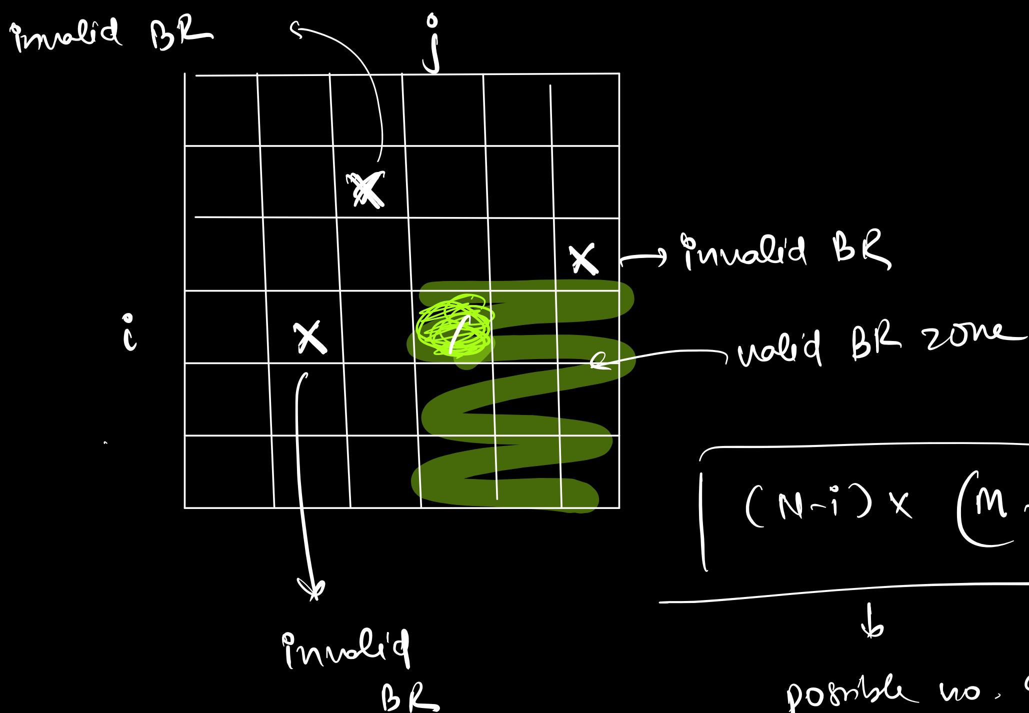
(TL, BR)



$\boxed{(i+1) \times (j+1)}$

$\downarrow$

cells can be a TL



$$\boxed{(N-i) \times (M-j)}$$

$\downarrow$

possible no. of valid BR.

$$\boxed{\begin{aligned} &\text{no. of submatrices containing cell }(i,j) = \text{all valid TL} \times \text{all valid BR} \\ &\#\text{count} = (i+1) \times (j+1) \times (N-i) \times (M-j) \end{aligned}}$$

contribution = arr[i][j] × count

do for all cells & add

$$\boxed{\begin{aligned} &TC \Rightarrow O(N \times M) \\ &SC \Rightarrow O(1) \end{aligned}}$$

Q3. Given a row wise & column wise sorted matrix (asc order)

Find max submatrix sum

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -20 & -16 & -4 & 8 \\ 1 & -10 & -8 & 12 & 14 \\ 2 & -1 & 6 & 21 & 30 \\ 3 & 5 & 7 & 28 & 42 \end{array}$$

Bruteforce

Try all possible submatrices

$O(N^2M^2)$

1D arr ⇒

$-6 \quad -1 \quad 2 \quad 4 \quad 8 \quad 9 \quad 10 \quad 12$

$\downarrow$

maxm element

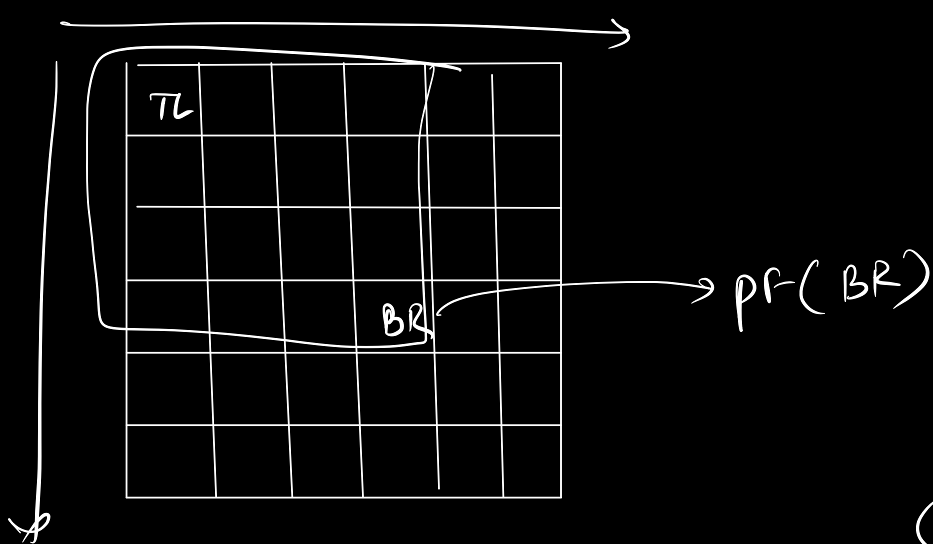
$\underline{\underline{N \times M}} \rightarrow$ maxm element ⇒ $N-1, M-1$

$$
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & -20 & -16 & -4 & 8 \\
1 & -10 & -8 & 12 & 14 \\
2 & -1 & 6 & 21 & 30 \\
3 & 5 & 7 & 28 & \boxed{42}
\end{array}
\quad \leftarrow BR
$$

BR is fixed → N-1, M-1

↓

only to find TL of submatrix (which will have max sum)



(N-1, M-1)

↑

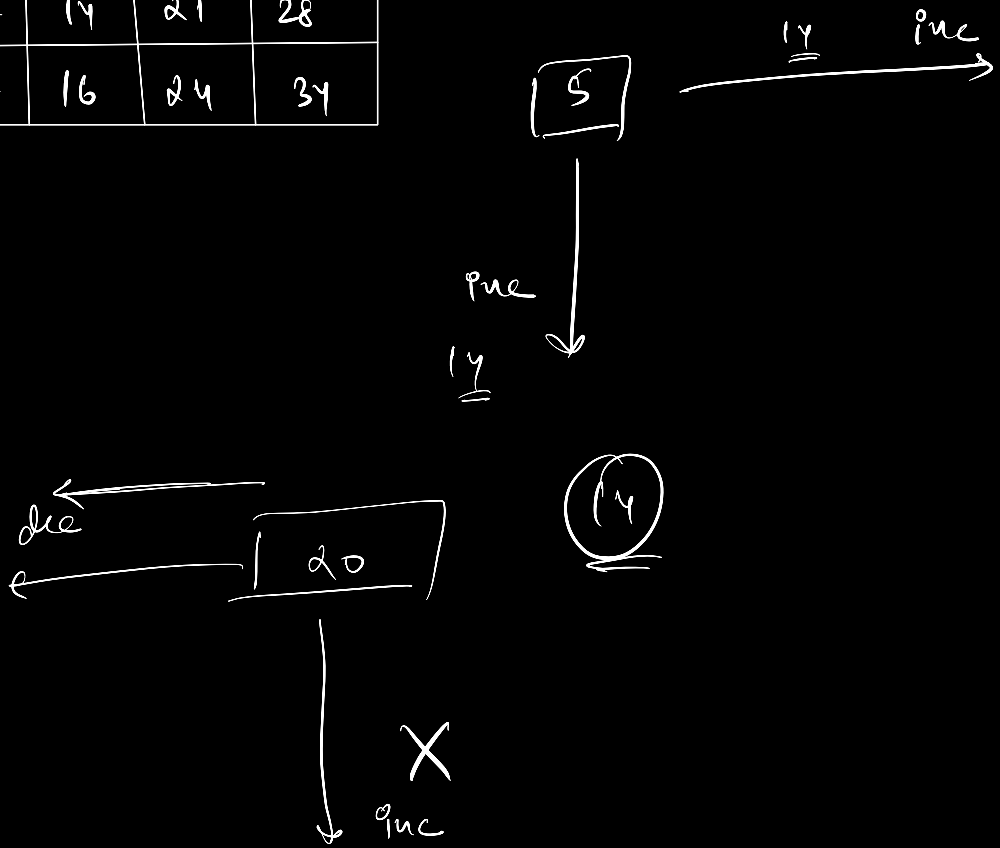
* take pfSum in reverse direction from BR and return (i,j) → TL where pfSum is maxm

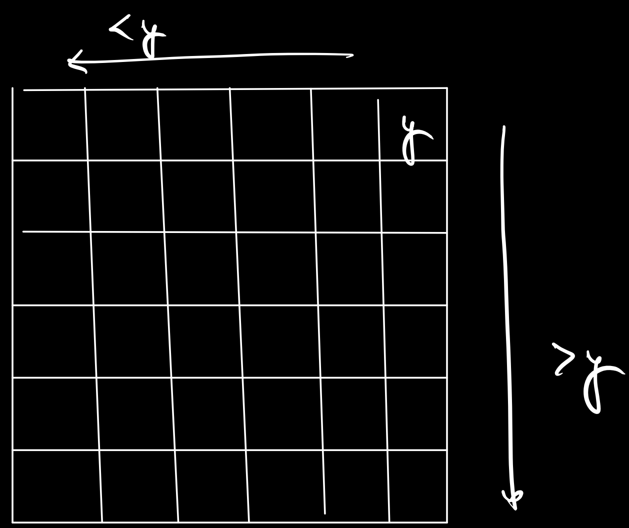
TC → O(N×M)

Q4. Given a matrix sorted row wise & colm wise, search if element 'k' is present.

	0	1	2	3
0	5	10	15	20
1	6	12	18	24
2	7	14	21	28
2	8	16	24	34

K = 14





search (x)

x == y
return true

x < y
go left

x > y
go down

5	10	15	20
6	12	18	24
7	14	21	28
8	16	24	34

k = 14

29

TC = O(N+M)

every iteration we are rejecting a row or a column.