2D Array

Content

- Find in row wise & column wise sorted matrix
- Row with max no. of 1s
- Spiral Matrix
- Sum of all submatrices sum

Amreshwar
ANUBHAV RAMNANI
anurag jain
Arunava Basak
Bhavesh Pandey
Chandra Shekhar Bhatt
Deepshikha Arora
Dhasthagiri Reddy
Dhruv
Harikrishnan A
Harshad Sanjay Marathe
Induja
jeevanantham
Kanhu
M S Haseeb Khan
Mahesh Baswaraj
Manohar A N
MOHAMMAD ALI
Nikhil D
Pallavi V Rao
Piyush
Pranjul Kesharwani
Praveen Kumar
Priyank Varshney
Shambhavi
Shivam Shiv
Shivanand Patil
Simin Shaikh
Souban
Surabhi Kumari
Uddeepta Saikia
Ved Verma

Rules

1> Q ⟶ QT

2> A ⟶ PC

3> Always be active

Q> Given a row wise and col wise sorted matrix
Find out whether k is present or not.

I/p A =

-5	-2	1	13
-4	0	3	14
-3	2	6	18

k = 13 True

k = 2 True

k = -1 False

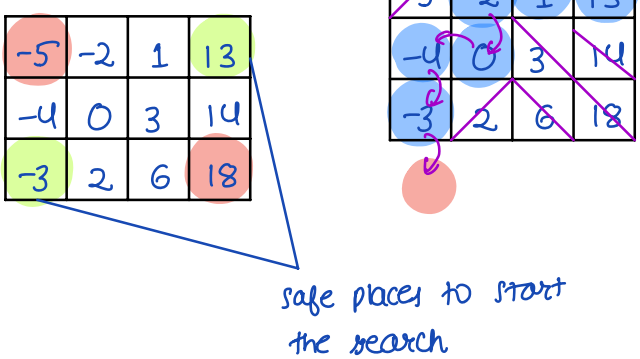
Bruteforce — Iterate over all the cells of the matrix

Best case — k is first cell. TC : O(1)

Worst case — k is not present. TC : O(R*C)

```
for r ⟶ 0 to R-1 {
    for c ⟶ 0 to C-1 {
        if(A[r][c] == k ) return true
    }
}
return false
```

k = -1



if k > current cell
down
else
left

Pseudo code

R = no of rows

C = no. of cols

```
// start from top right cell
r = 0      c = C-1

while ( r < R && c >= 0 ) {
    if (A[r][c] == k)  return true
    if ( A[r][c] < k ) {
        r += 1
    } else {
        c -= 1
    }
}
return false
```

k = 3

-5	-2	1	13
-4	0	3	14
-3	2	6	18

k = -4

-5	-2	1	13
-4	0	3	14
-3	2	6	18

k = 6

-5	-2	1	13
-4	0	3	14
-3	2	6	18

TC: O(R+C)

SC: O(1)

all rows are sorted

Q> Given a binary sorted matrix A[N][N]
Find the row with max # of 1's

NOTE
- If two rows have the max no. of 1, return lower index
- Assume each row to be sorted by values.

I/p A =

0	0	1	1
1	0	0	1
2	0	1	1

output : 0

I/p A =

0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

output : 3

Bruteforce ⟶ For every row ⟶ cnt # 1s
return the row with max # 1s

```
for r ⟶ 0 to R-1 {
    cnt = 0
    for c ⟶ 0 to C-1 {
        cnt += A[r][c]
    }
    if (cnt > mcnt) { mcnt = cnt, ans = r }
}
```

TC : O(R*C)
SC : O(1)

A =

	0	1	2	3	4	5
0	0	0	0	0	1	1
1	0	0	1	1	1	1
2	0	0	0	0	0	1
3	0	0	0	0	1	1
4	0	1	1	1	1	1
5	0	0	0	1	1	1

ans = 0 0 0 1 4

If (A[r][c] == 1)
← left
else
↓ down

Pseudocode

```
r=0     c = c-1
ans = 0
while (r < R && c >=0) {
    // keep moving left when val ==1
    while (c >=0 && A[r][c] == 1) {
        ans = r
        c--
    }
    r++
}
```

TC: O(R+C)

SC: O(1)

Print Boundary Elements ****

Given a matrix of N * N, Print boundary elements in clockwise direction.

I/p A =

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Output : 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Pseudocode

```
void printBoundary ( N , r , c ) {

    r = 0    c = 0

    // N-1 value in top row
    for i ⟶ 1 to N-1 {    // N-1 iteration
```

```
        print (A[r][c])
        c++
    }

// N-1 values in right col
for i ⟶ 1 to N-1 {   // N-1 iteration
    print (A[r][c])
    r++
}

// N-1 values in bottom row
for i ⟶ 1 to N-1 {   // N-1 iteration
    print (A[r][c])
    c--
}

// N-1 values in left col
for i ⟶ 1 to N-1 {   // N-1 iteration
    print (A[r][c])
    r--
}

// edge case
if (N == 1) {
    print (A[r][c])
}
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

TC : O(N)

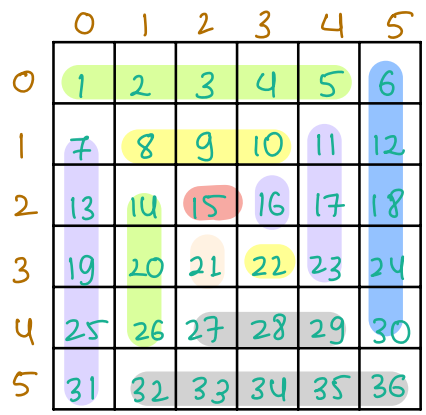
SC : O(1)

Spiral Matrix ****

Given a matrix of N * N.
Print elements in spiral order in clockwise direction.

A =

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36



Pseudocode

```
r = 0   c = 0
while (N > 0) {
    printBoundary(N, r, c)
    // update N, r, c
    N = N - 2
    r += 1
    c += 1
}
```

TC: $O(N^2)$
SC: $O(1)$

Break : 22:50

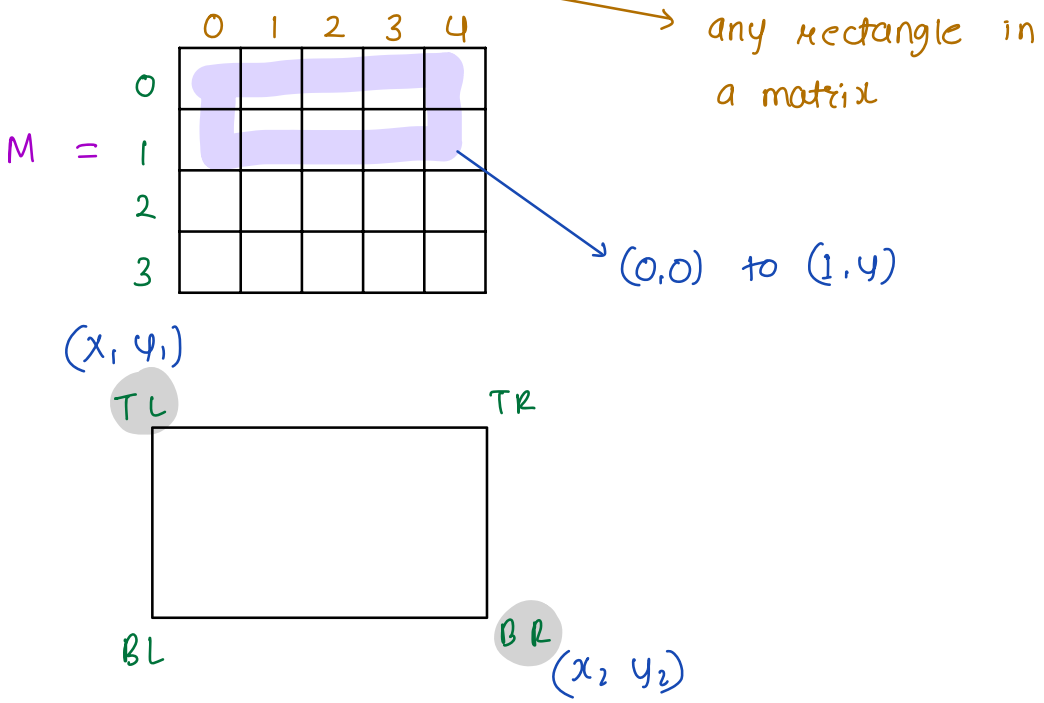
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

N	r	c
5	0	0
3	1	1
1	2	2

what is a submatrix ?

Subarray ⟶ continuous part of an array

Submatrix ⟶ continuous part of a matrix



Sum of all submatrices sum

Given a matrix A[R][C]. Determine the sum of all possible submatrices.

M =

	0	1	2
0	4	9	6
1	5	-1	2

All possible submatrices

size = 1

4	4
9	9
6	6
5	5
-1	-1
2	2

size = 2

4 9	13	
9 6	15	
5 -1	4	
-1 2	1	
4 5	9	
9 -1	8	
6 2	8	

size = 3

4 9 6	19
5 -1 2	6

size = 4

4 9 5 -1	17
9 6 -1 2	16

4 9 6 5 -1 2	25

output = 166

$$\sum_{rc} \#\text{submatrices } A[r][c] \text{ present} * A[r][c]$$

Brute force

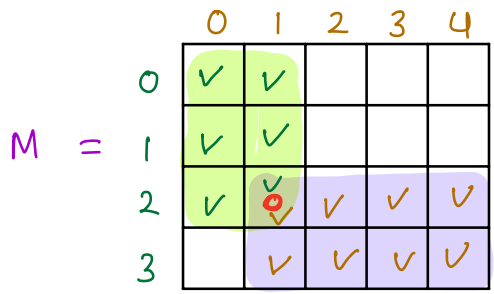
```
// Figure out all posible comb. of TL & BR        (r1, c1)  (r2, c2)
ans
for r1 ---> 0 to R-1 { } TL
    for c1 ---> 0 to C-1 {
        for r2 ---> r1 to R-1 { } BR
            for c2 ---> c1 to C-1 {
                sub = 0
                for r ---> r1 to r2 {
                    for c ---> c1 to c2 {
                        sub += A[r][c]
                    }
                }
                ans += sub
            }
    }
```

TC : $O((R*C)^3)$

SC : $O(1)$

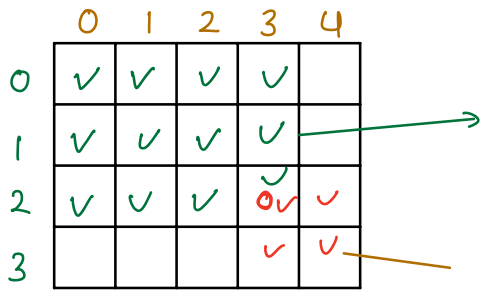
N * N matrix

TC : $O(N^6)$

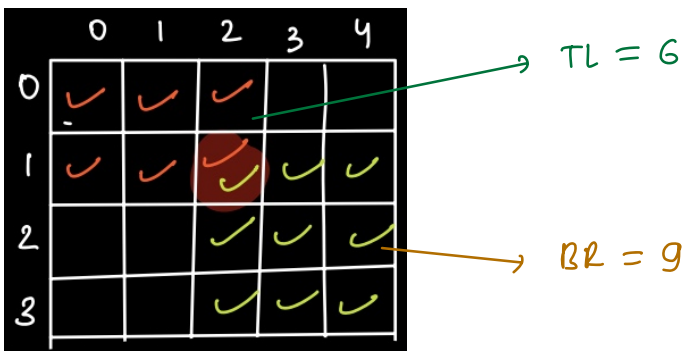


✓ TL

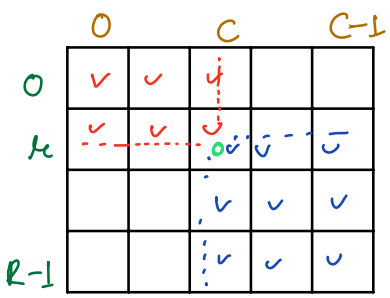
✓ BR



# of submatrices containing red dot = 12 * 4 = 48



9 * 6 = 54



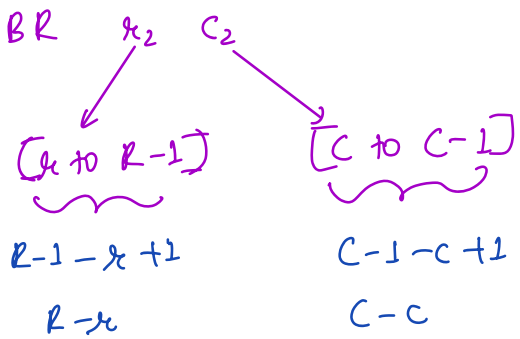
TL (r_1, c_1)

$[0, r]$ $r+1$

$[0, c]$ $c+1$

# of way to select TL = $(r+1) * (c+1)$

# of submatrices containing (r, c) = #TL * #BR



# of ways to select BR = (R-r) * (C-c)

contribution of A[r][c] = A[r][c] * # sub matrices containing A[r][c]

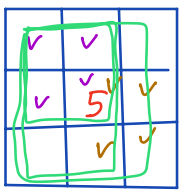
= A[r][c] * #TL * #BR

Pseudocode

```
ans = 0                                   → long
                                          TC: O(R*C)
                                          SC: O(1)
for r ⟶ 0 to R-1 {
    for c ⟶ 0 to C-1 {
        TL = (r+1) * (c+1)                long
        BR = (R-r) * (C-c)                long
        ans += A[r][c] * TL * BR
    }
}
print(ans)
```

Doubt session

8



5 * 16