Today's Agenda:

Starting 7:05

1. Search in a row-wise & column-wise sorted matrix.
2. Row with maximum ones.
3. Printing boundary of a matrix.
4. Spiral Matrix
5. Sum of all submatrices sum.

1. Given a row wise & column wise sorted matrix. Check whether element K is present or not.

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

K	Output
13	true
0	true.
5	false.

Brute Force

Iterate over every cell.

Q-1.

$N * M$

Optimisation

0 = 0.

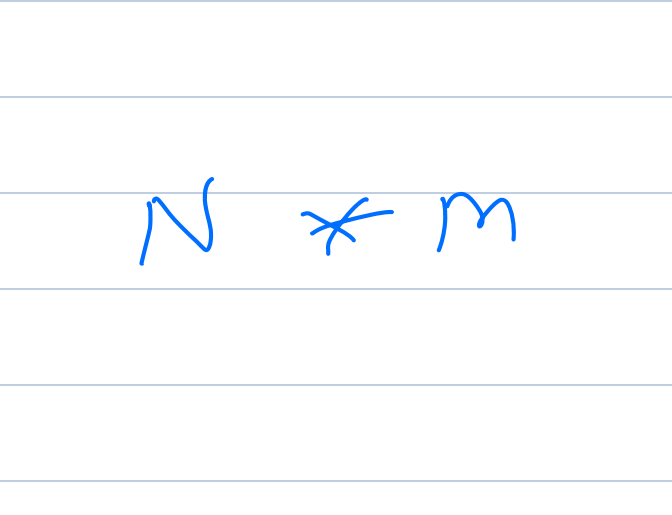
Searching → 0

-2 < 0

-5 < 0

13 > 0

1 > 0

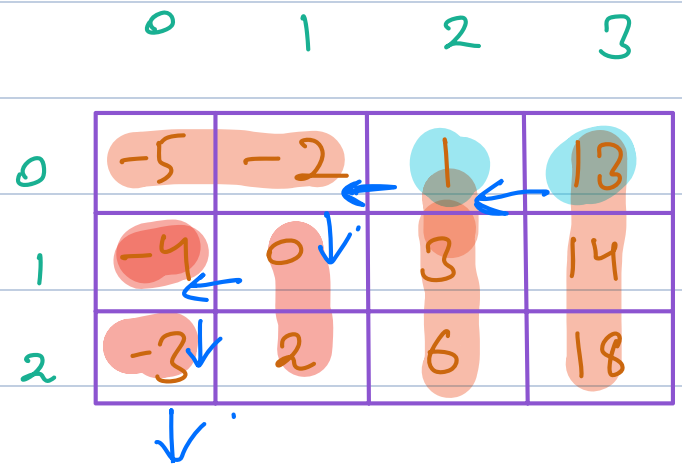


0 7-1

-1.

-4 < -1

-3 < -1



13 > -1

1 > -1

-2 < -1

Approach

① Start at top right cell. or bottom left

reject row

② If A[i][j] < k, move ↓ else move ← (reject col)

③ Repeat until element is found or (i || j) goes out of bound.

Code

T.C → O(N+m)

```
int i=0, j = m-1;
while (i < N && j >=0) {
    if (arr[i][j] == K)
        return true;
    else if (arr[i][j] < K) {
        i++;
    }
    else {
        j--;
    }
}
```

2. Given a binary sorted matrix A of size N * N. Find the row with maximum number of 1.

Note:- ① If two rows have same ans, return the one having lower index.

② Every row is sorted.

Ex:- 1

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

0.

Ex: 2

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

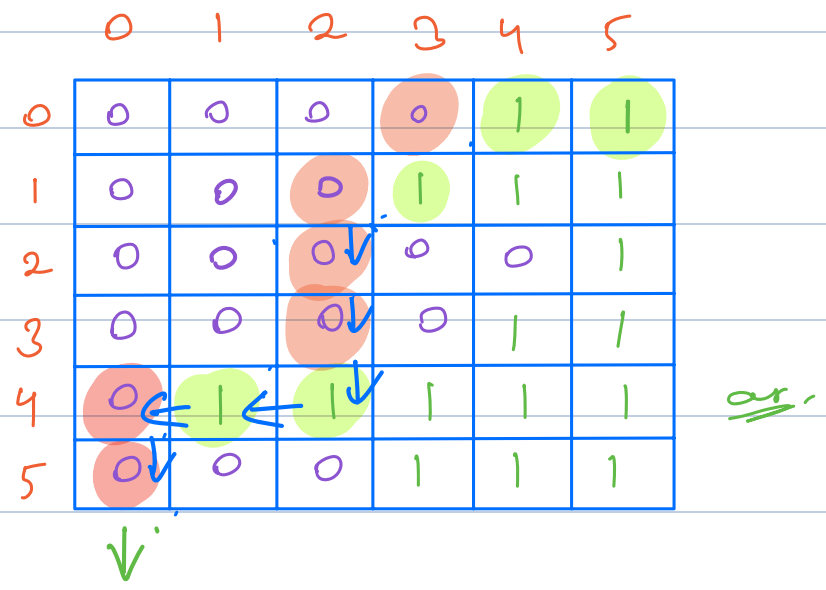
3.

Quiz 3

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

2.

Optimisation



or.

```
i=0; j = m-1, ans = -1;
while (i<N && j>=0) {
    while ( j>=0 && arr[i][j] == 1
    { j--;
      ans = i;
    }
    i++;
}
return ans;
```

i=0; j = 5; ans = 0;
x 4
x 3 4 5 6 3 x 0

Algorithm

1. Start at i=0; j = m-1;
2. If 1 is present, move left (decrement j) & update ans to curr row.
3. If 0 is present, move down (incr i).

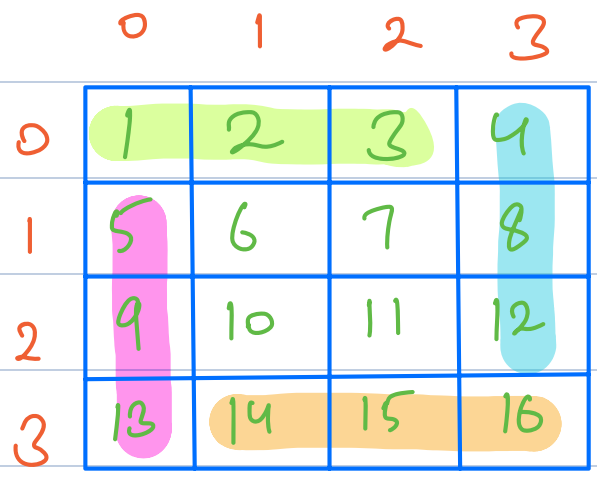
Code

```
i=0, j = m-1;   ans=-1;

while (i<N && j>=0) {

    while ( j>=0 && arr[i][j]==1)
        { j--;
          ans = i;
        }

    i++;
}

return ans;
```

T.C → O(N+m).

S.C → O(1)

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



Output :- 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5

Approach :

Print N-1 elements in each :-

a) First Row →
b) Last Column ↓
c) Last Row ←
d) First Column ↑

Code:-

```
void printBoundaryElements(int A[][]) {
    i = 0; j = 0;
```

① N-1 in the first row →

```
    for (it = 1; it < N; it++) {
        print(a[i][j]); j++;
    }
```

② N-1 in the last column ↓

```
    for (it = 1; it < N; it++) {
        print(a[i][j]); i++;
    }
```

③ N-1 in the last row ←

```
    for (it = 1; it < N; it++) {
        print(a[i][j]); j--;
    }
```

④ N-1 in the first column ↑

```
    for (it = 1; it < N; it++) {
        print(a[i][j]); i--;
    }
}
```

T.C → O(N);
S.C → O(1);

4. Given an integer A, generate a square matrix of size A * A filled with elements in spiral order from 1 to A^2.

A*A.

1 to 16

Ex:- A = 4.

Output :-

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

Ex:- A = 6

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

8:35.

i	j	A.
0	0	6
		-2
1	1	4.
2	2	2.
3	3	0

Iterate 5 times in 4 direction

Iterate 3 times in 4 direction.

Iterate 1 times in 4 direction

Code:

int i=0, j=0, A ; ← known.
, cout=0;
int [A][A];

while (A > 1) {

① A-1 in the first row →

for (it = 1 ; it < A ; it++) {
A[i][j] = ++cout; j++;
}

② A-1 in the last column ↓

for (it = 1 ; it < A ; it++) {
A[i][j] = ++cout; i++;
}

③ A-1 in the last row ←

for (it = 1 ; it < A ; it++) {
A[i][j] = ++cout; j--;
}

④ A-1 in the first column ↑

for (it = 1 ; it < A ; it++) {
A[i][j] = ++cout; i--;
}

i++; j++; A -= 2;

if (A == 1) {
A[i][j] = ++cout;
}

}

A = 5

	1	2	3	4
0	1	2	3	4	5
1	16	17	18	19	6
2	15	24	25	20	7
3	14	23	22	21	8
4	13	12	11	10	9

i, j A

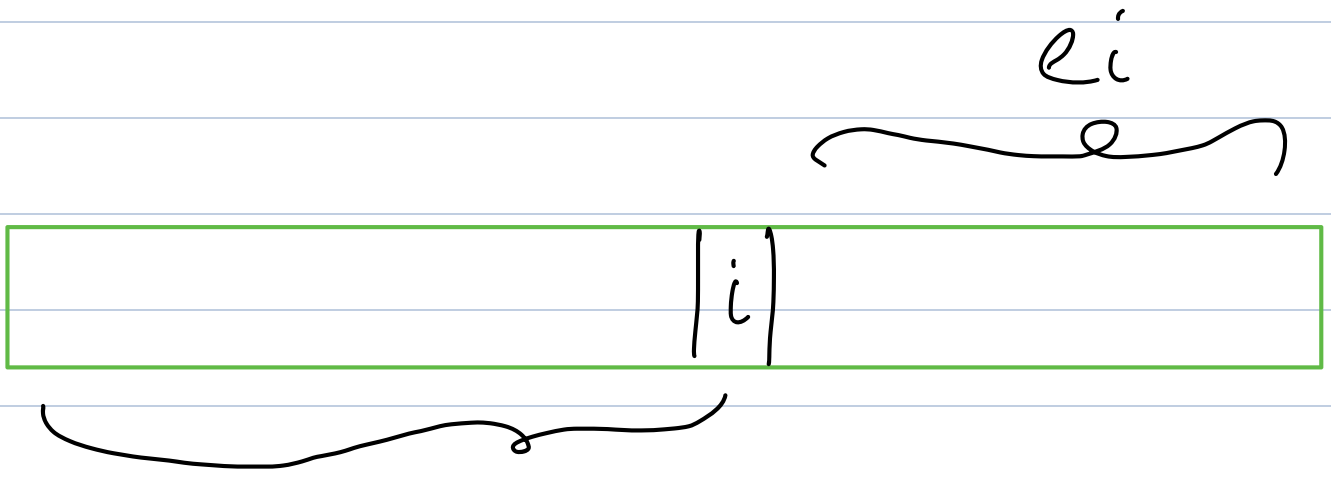
0, 0 5

1, 1 3

2, 2 1

3, 3 -1

TC → $O(A^2)$.



(i+1) * (N-i)

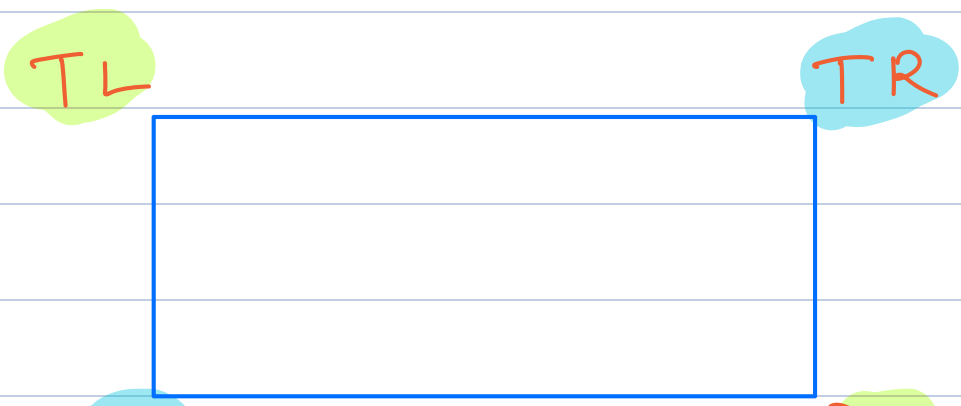
Submatrices

→ Same concept as a subarray.

→ A continuous part of a 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

Uniquely identifying a rectangle



Either TL & BR

OR BL & TR.

5. Given a matrix. Find the sum of all submatrix sums.

1 * 3 * 2 * 1

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

All submatrices

1 x 1

4 -4

9 -9

6 -6

5 -5

-1 --1

2 -2

1 x 2

4, 9 -13

9, 6 -15

5 -1 -4

-1, 2 -1

1 * 3

4, 9, 6 -19

5, -1, 2 -6

2 * 1

4 -9

5

9 -8

-1

6 -8

2

2 * 3

25

2 x 2

4 9 -17 (17)

5 -1

9 6 16

-1 2

166

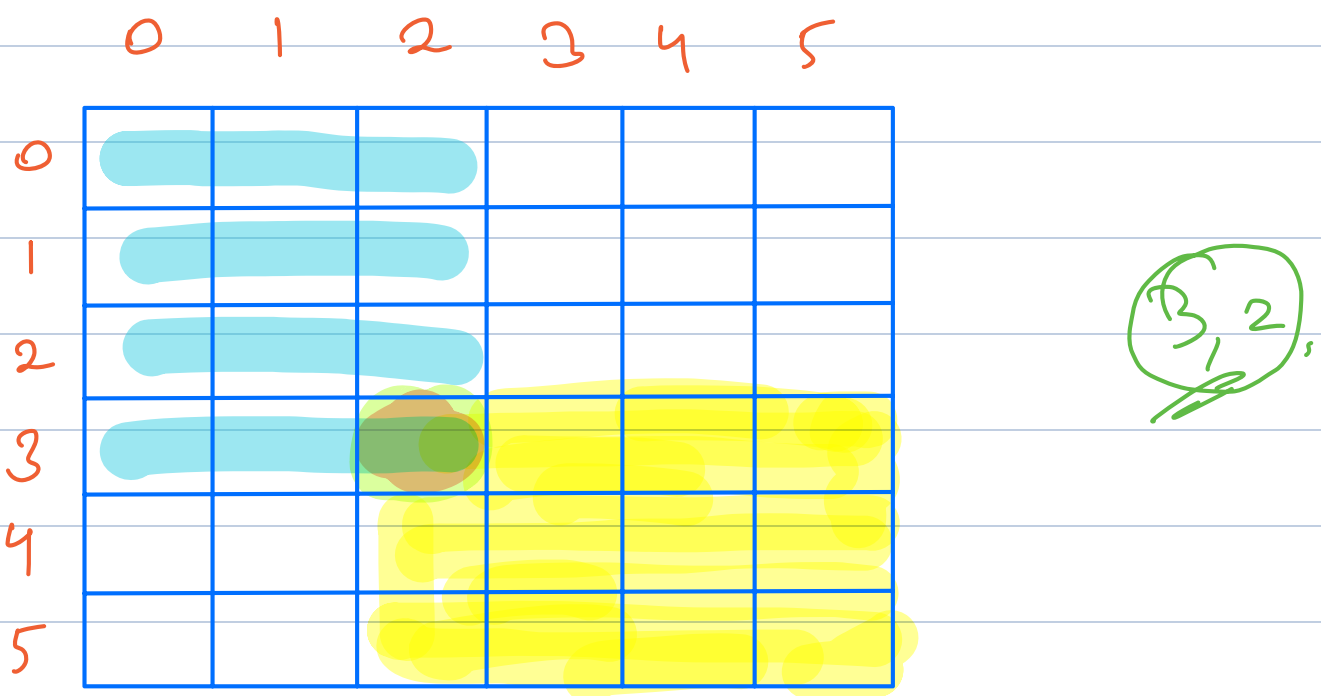
Total Submatrices → $O(N^4)$

Iterate on next submatrix

$O(N^2)$

$O(N^6)$

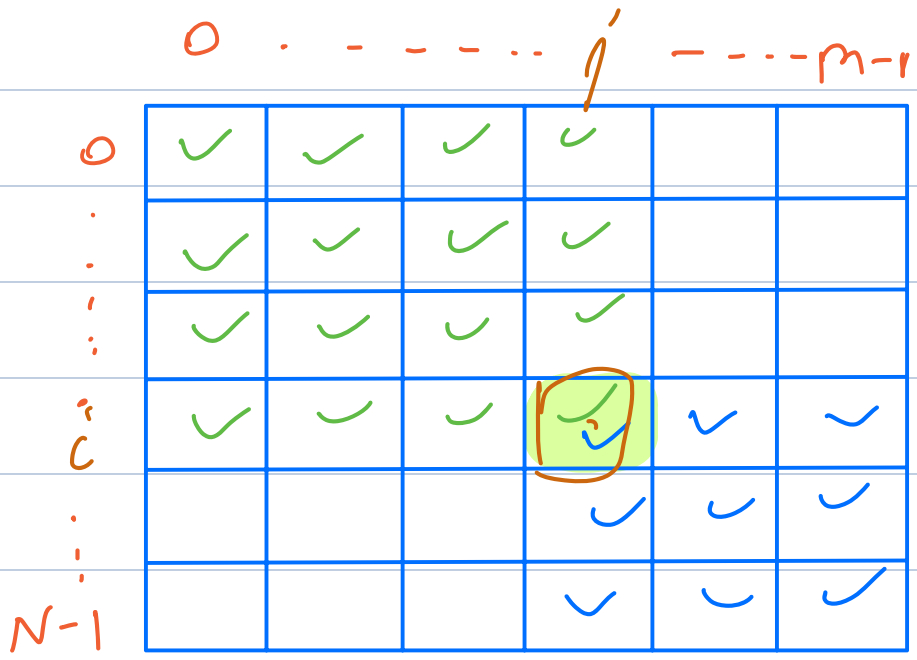
Approach



12 * 12 = 144

Quiz

6 * 9 = 54



(0, j)

j - 0 + 1

(i, N-1)

(j, m-1)

T.L ⟶ (i+1) * (j+1).

B.R ⟶ (N-i) * (m-j).

Contribution of element at index i, j

= (i+1) * (j+1) * (N-i)
* (m-j) * A[i][j];

Code:

```
ans = 0;

for (i=0; i<N; i++) {
    for (j=0; j<m; j++) {
        ans += (i+1) * (j+1) * (N-i)
               * (m-j) * A[i][j];
    }
}
return ans;
```

T.C → O(N*m).