Lecture :- Arrays 2

Agenda

- find k in row and col wise sorted matrix.
- Row with maximum number of 1's.
- Boundary print of matrix
- Spiral order of a matrix
- Sum of all submatrices sum.

Qu: Given a row and col wise sorted matrix. find out whether 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 = 13 (true)

k = 2 (true)

k = 15 (false)

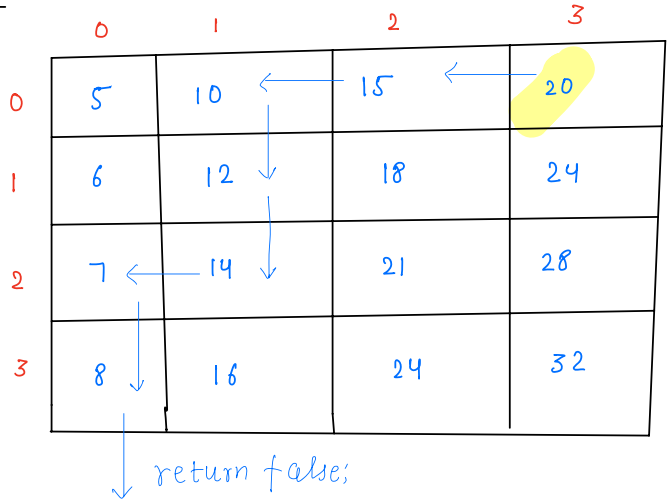
Brute force:

```
int findkinsortedMatrix(int[][] mat) {
    int n = mat.length;
    int m = mat[0].length;
    for(i=0; i<n; i++) {
        for(j=0; j<m; j++) {
            if(mat[i][j] == k) {
                return true;
            }
        }
    }
    return false;
}
```

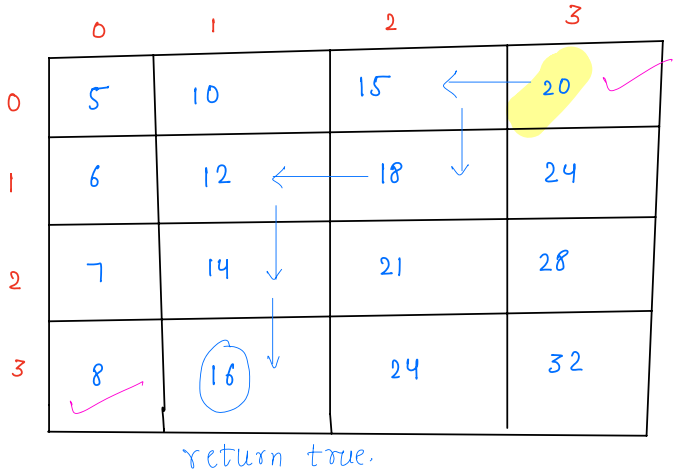
TC: O(n*m)

SC: O(1)

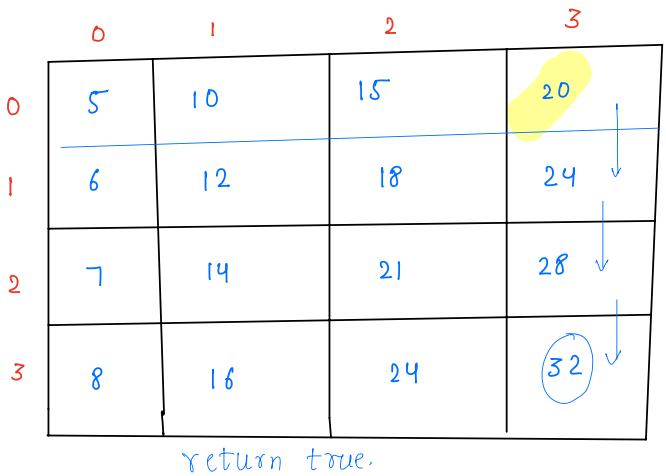
Approach 2



k = 13



k = 16



k = 32

Algorithmic code

```
boolean search( int[][] mat, int k) {
        int n = mat.length;
        int m = mat[0].length;
        int i = 0;
        int j = m-1;
        while( i<n  &&  j>=0) {
                if( mat[i][j] == k) {
                        return true;
                }
                if( mat[i][j] < k) {
                        i++;
                }
            else if( mat[i][j] > k) {
                        j--;
                }
        }
        return false;
}
```

TC: O(n)

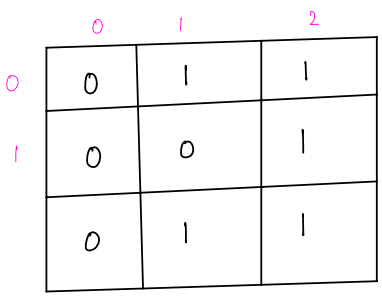
SC: O(1)

Problem 2

Given a binary (row) sorted matrix of size n * n. find row with maximum number of 1.

Note. 1.) If two rows have maximum no. of 1 then return the row with lower idx.

2.) Assume each row is sorted by values



ans = 0

0	0	0	0
0	0	0	1
0	0	1	1
0	1	1	1

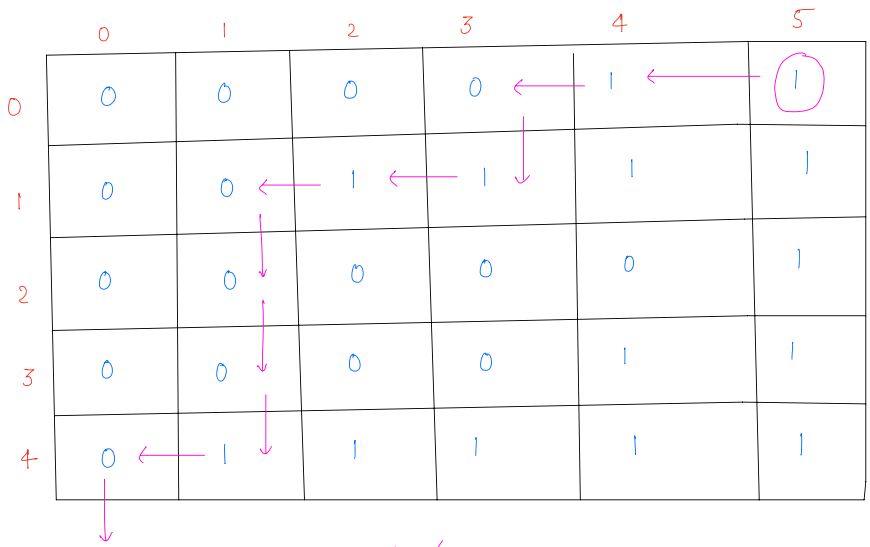
3rd row

Brute force:

TC: O(n * n)

SC: O(1)

Approach 2



ans = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

Algorithmic code

```
int maxOnesRow(int[][] matrix) {
    n = mat.length;
    i=0;  j=n-1;
    while(i<n && j>=0) {
        while(j>=0 && mat[i][j]==1) {
            j--;
            ans=i;
        }
        i++;
    }
    return ans;
}
```

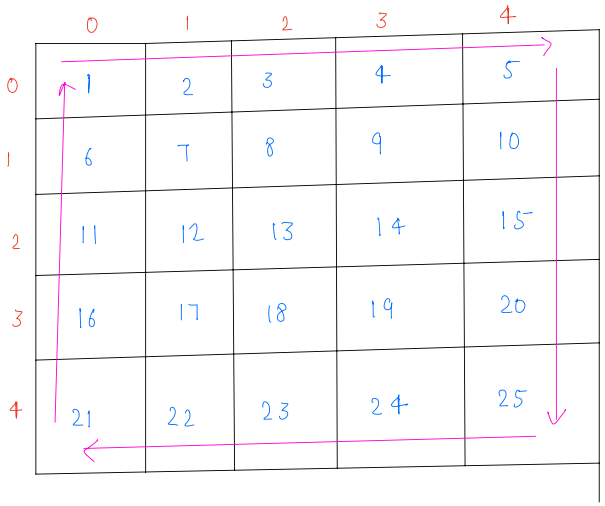
TC: O(n)

SC: O(1)

Problem 3

Boundary level printing

Given mat[n][n], print boundary elements in clockwise direction.

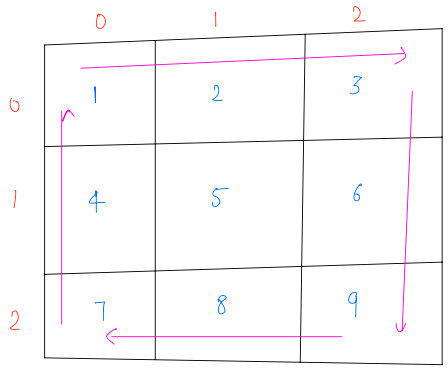


output

1 2 3 4 5

10 15 20 25 24

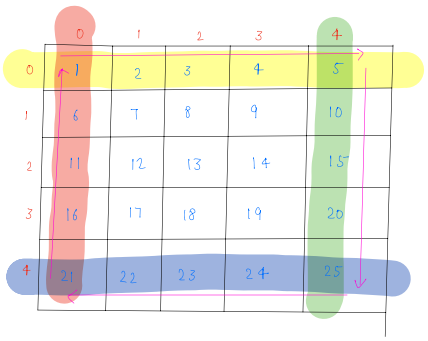
23 22 21 16 11 6



output

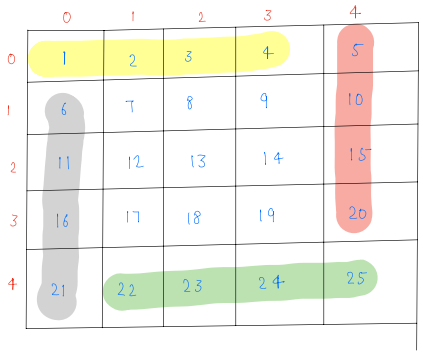
1 2 3 6 9

8 7 4



—— wrong (Repetition)

Approach



$i = 0$ and $j = 0$

i	j	
0	0	↓ +1
0	1	↓ +1
0	2	↓ +1
0	3	
0	4	(stop)

(Yellow)

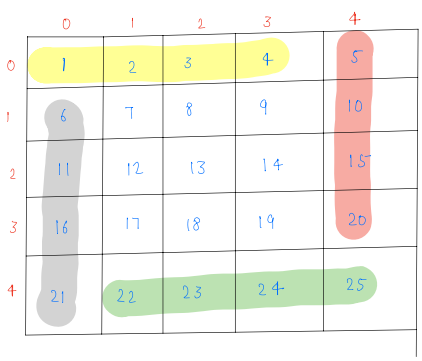
j++

$i = 0$ and $j = 4$

0	4
1	4
2	4
3	4
4	4 (stop)

(Red)

i++



$i = 4$ and $j = 4$

4	4
4	3
4	2
4	1
4	0 (stop)

Green

j--

$i = 4$ and $j = 0$

4	0
3	0
2	0
1	0
0	0 (stop)

(Grey)

i--

Algorithmic code

```
void boundaryPrint(int[][] mat) {
    n = mat.length;
    i = 0;
    j = 0;

    // 0th row.
    for(k=0; k<n-1; k++) {
        print(mat[i][j]);
        j++;
    }

    // Last column.
    for(k=0; k<n-1; k++) {
        print(mat[i][j]);
        i++;
    }

    // Last row.
    for(k=0; k<n-1; k++) {
        print(mat[i][j]);
        j--;
    }

    // first column
    for(k=0; k<n-1; k++) {
        print(mat[i][j]);
        i--;
    }
}
```

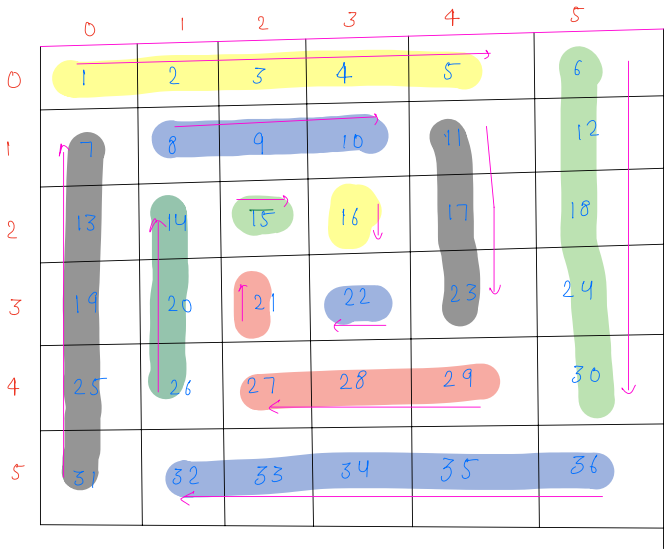
TC: $O(n) + O(n) + O(n) + O(n) \simeq O(n)$

SC: $O(1)$

H|W Extend mat[n][m].

Spiral order matrix

Given mat[n][n], print spiral order of matrix in clockwise direction.

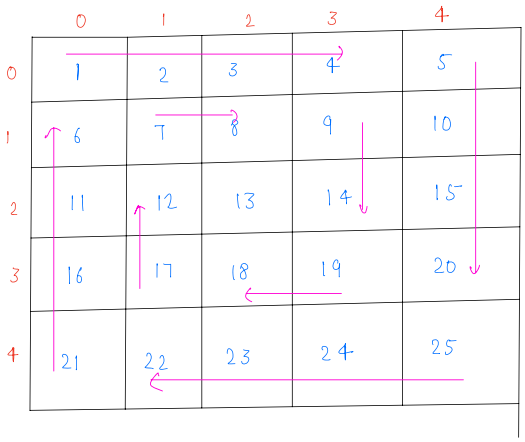


output

i	j	(k) Loop print
0	0	5 (n-1)
1	1	3 (n-3)
2	2	1 (n-5)

Approach

	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



i	j	K (Loop print)
0	0	4 (n-1)
1	1	2 (n-3)
2	2	0 (n-5)

```
if (loop print == 0) {
print(mat[i][j]).
}
```

Algorithmic code

```
void spiralorder(int[][] mat) {
    n = mat.length;
    i=0; j=0; loopprint = n-1;
    while(loopPrint >=0)
        if(loopprint ==0) {
            print(mat[i][j];
            break;
        }
        // 0th row.
        for(k=0; k< loopprint ; k++) {
            print(mat[i][j]);
            j++;
        }
        // Last column.
        for(k=0; k< loopprint k++) {
            print(mat[i][j]);
            i++;
        }
        // Last row.
        for(k=0; k< loopprint ; k++) {
            print(mat[i][j]);
            j--;
        }
        // first column
        for(k=0; k< loopprint ; k++) {
            print(mat[i][j]);
            i--;
        }
    i++;
    j++;
    loopPrint -=2;
}
```

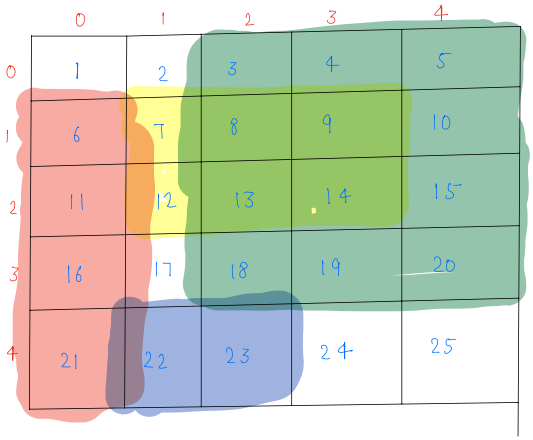
TC: $O(n^2)$
SC: $O(1)$

Break: 8:43 - 8:53

Submatrices and its identification

Subarray is continuous and ordered part of array

Submatrix is " " " " " matrix.



Submatrices

1	2	3	4

Subarrays:

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

Problem 5

Sum of all submatrices sum

Given mat[n][m], determine sum of all the possible submatrices.

mat[][]

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

All submatrices

[4] = 4

[9] = 9

[6] = 6

[5] = 5

[-1] = -1

[2] = 2

[4 9] = 4+9

[4 9 6] = 4+9+6

$\begin{bmatrix} 4 & 9 \\ 5 & -1 \end{bmatrix}$ = 4+5+9-1

$\begin{bmatrix} 4 & 9 & 6 \\ 5 & -1 & 2 \end{bmatrix}$ = 4+9+6+5-1+2

[9 6] = 9+6

$\begin{bmatrix} 9 & 6 \\ -1 & 2 \end{bmatrix}$ = 9+6-1+2

$\begin{bmatrix} 4 \\ 5 \end{bmatrix}$ = 4+5

$\begin{bmatrix} 9 \\ -1 \end{bmatrix}$ = 9-1

$\begin{bmatrix} 6 \\ 2 \end{bmatrix}$ = 6+2

[-1 2] = -1+2

[5 -1] = 5-1

[5 -1 2] = 5-1+2

Approach

sum = 4 * 6 + 9 * 8 + 6 * 6 + (-1) * 8 +

2 * 6 + 5 * 6

sum = mat[0][0] * occ1 +

mat[0][1] * occ2 +

mat[0][2] * occ3

⋮

mat[n-1][m-1] * $occ_{n-1,m-1}$

mat[][]	0	1	2
0	4	9	6
1	5	-1	2

Challenge: find the occurence.

mat[][]	0	1	2
0	4	9	6
1	5	-1	2

Occ of 9 ?

No of start cells for which 9 is part of ?

↳ 4, 9 ⟹ x

No of end cells for which 9 is part of ?

↳ 2, 6, -1, 9 ⟹ y

Occurence = x * y

[9] ✓

[4 9] ✓

[4 9 6] ✓

[4 9
5 -1] ✓

[4 9 6
5 -1 2] ✓

[9 6] ✓

[9 6
-1 2] ✓

[9
-1] ✓

h/w: x = ? ⟶ (i+1) * (j+1)

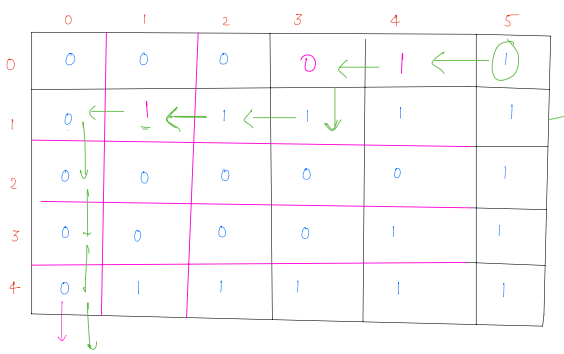
y = ? ⟶ (n-i) * (m-j)

```
int totalSubmatricesSum(mat[][]) {
    total = 0;
    n = mat.length;
    m = mat[0].length;
    for(i=0; i<n; i++) {
        for(j=0; j<m; j++) {
            x = (i+1) * (j+1);
            y = (n-i) * (m-j);
            total += mat[i][j] * x * y;
        }
    }
    return total;
}
```

TC: O(n*m)

SC: O(1)

Doubt



ans = ~~0~~ 1

j = ~~4 3 2 1~~ 0

i = ~~0 1 2 3~~ 4

```
int maxOnesRow(int[][] matrix) {
    n = mat.length;
    i = 0;      j = n-1;
    while(i<n && j>=0) {
        while(j>=0 && mat[i][j] == 1) {
            j--;
            ans = i;
        }
        i++;
    }
    return ans;
}
```