Today's Agenda :-

→ 2d Matrices

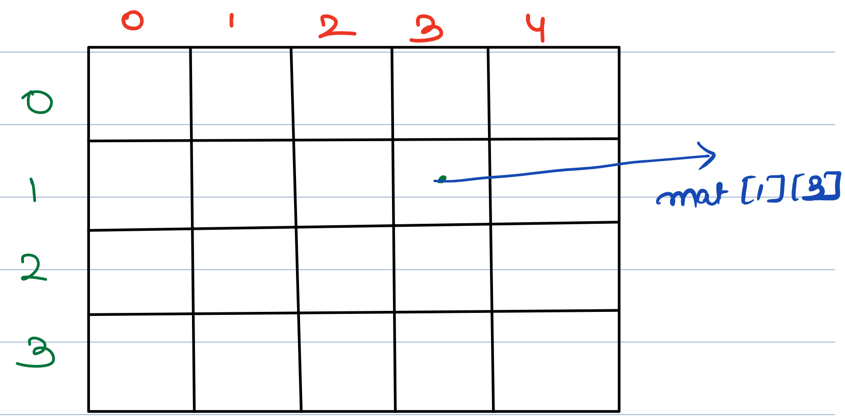
- Print row wise
- Print max column sum
- Print mat[N][N] diagonals
- Print mat[N][M] , R-L diagonals
- Transpose
- Rotate 90' clockwise
- Print all boundaries in clockwise

Quote :-

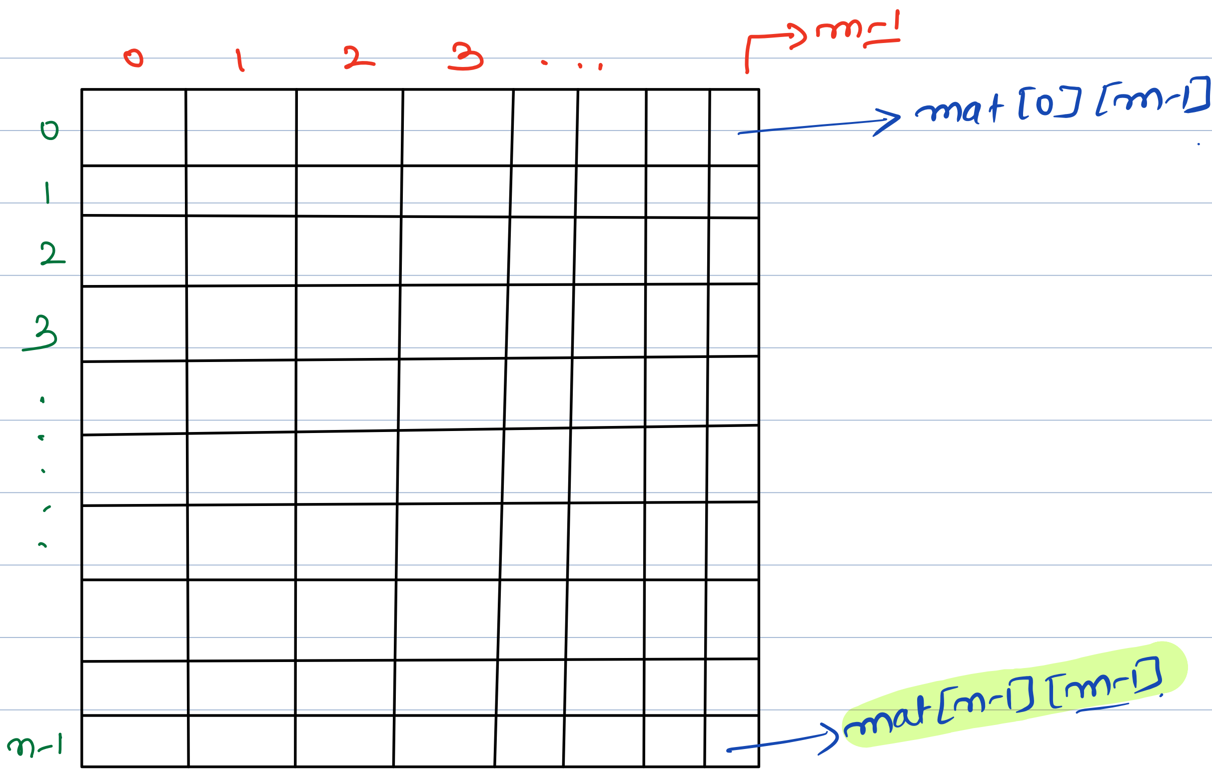
cols

int mat[4][5]

row



int mat[N][M]



1Q) Given mat[N][M], print row-wise

mat[3][4]

	0	1	2	3
0	3	8	9	2
1	1	2	3	6
2	4	10	11	17

```
for(i=0; i<n; i++){
    for(j=0; j<m; j++){
        print(arr[i][j])
    }
    print("\n");
}
```

output → 3 8 9 2
1 2 3 6
4 10 11 17

i	j
0 →	0
	1
	2
	3
1 →	0
	1
	2
	3

T.C → O(n * m)

S.C → O(1).

2Q) Given mat[N][M], find max column sum

MaxSum = Integer.MIN_VALUE

	0	1	2	3
0	3	8	9	2
1	1	2	3	6
2	4	10	11	8
	8	20	23	16

Ans.

```
for(j=0; j<m; j++){
    int sum=0;
    for(i=0; i<n; i++){
        sum+= Mat[i][j];
    }
    maxSum = Max(maxSum, sum)
}
```

sum = 0

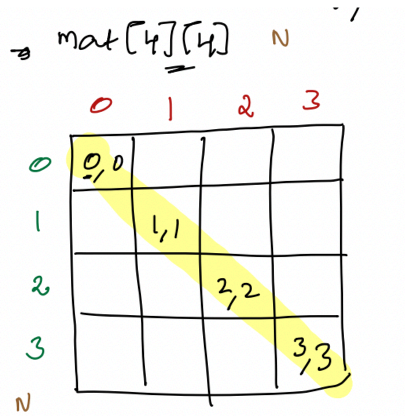
maxSum = 20

j	i
0	0, 1, 2
1	0, 1, 2

T.C → O(M*N)

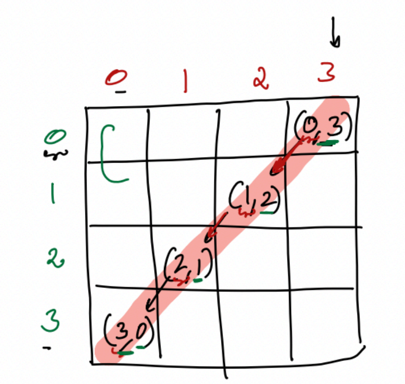
S.C → O(1)

Ques 3) Given a mat[N][N], print diagonals → left to right
→ right to left.




```
i=0,
while (i<N) {
    print(mat[i][i]
    i++
}
```

T.C → O(N)




```
i=0, j=n-1
while (i<n && j>=0)
    print(mat[i][j]);
    i++, j--
}
```

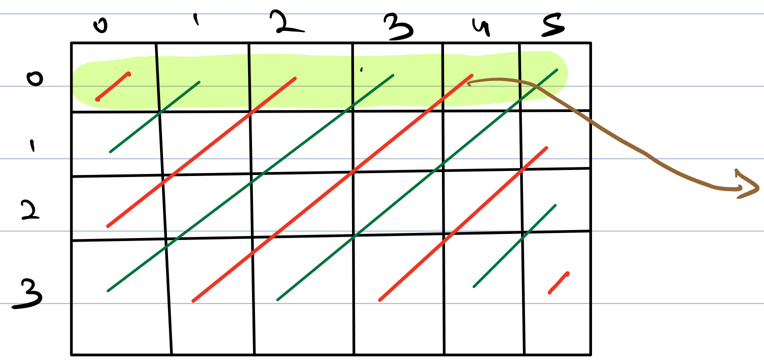
T.C → O(N)

S.C → O(1)

Ques) Given mat[N][M], print all diagonals going R-L.

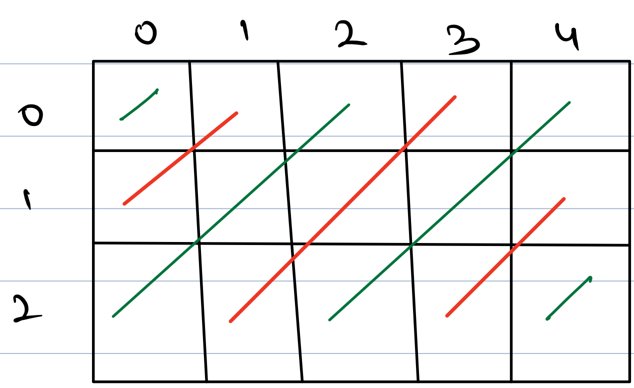
mat[4][6], diagonals = 9

no. of diagonals = M + N - 1



(0,4)
↓
(1,3)
↓
(2,2)
↓
(3,1)
↓
(4,0)
↓
row ended.

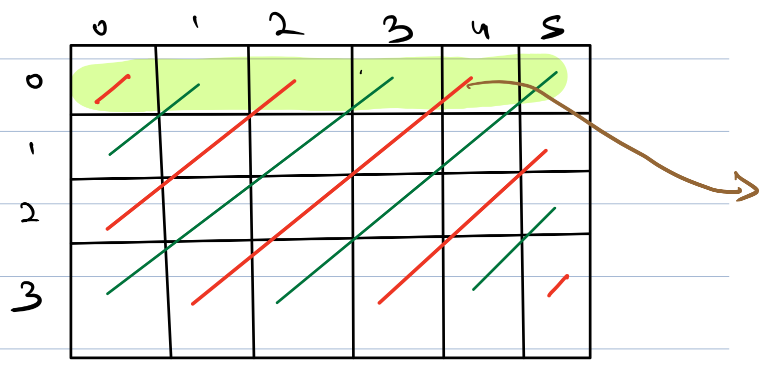
mat[3][5], diagonals = 7.



```
for (j=0; j<M ; j++) {
        x = 0, y = j
    while (x<n && y >=0 )
        Print (mat[x][y]) ;
        x ++ , y --
    }
}
```

T.C → O(M * N) .
S.C → O(1)

```
for (i=1; i<N; i++) {
        x = i, y = M-1
    while (x<n && y >=0 )
        Print (mat[x][y]) ;
        x ++ , y --
    }
}
```



5Q) Given a mat[N][N] find the transpose inplace

i) Given input mat[][] should update, SC: O(1)

mat[5][5]

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

Transpose

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

Approach :-

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

mat[0][1]
↓↑
mat[1][0]

mat[2][0]
↓↑
mat[0][2]

mat[3][0]
↓↑
mat[0][3]

mat[i][j] ⇄ mat[j][i]

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        if (i<j) {
            swap (arr[i][j], arr[j][i])
```

3

3

2

T.C → $O(n^2)$

S.C → $O(1)$

mat [3][4] $\xrightarrow{\text{after}}$ [4 x 3] we need a matrix

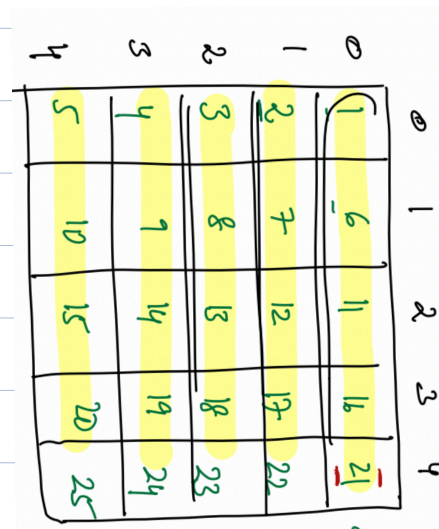
$$
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 1 & 2 & 3 & 4 \\
1 & 5 & 6 & 7 & 8 \\
2 & 9 & 10 & 11 & 12
\end{array}
\qquad \longrightarrow \qquad
\begin{bmatrix}
1 & 5 & 9 \\
2 & 6 & 10 \\
3 & 7 & 11 \\
4 & 8 & 12
\end{bmatrix}
$$

Break 10:04 – 10:14 pm.

Ques)

6Q) Given a mat[N][N] rotate 90° Clockwise, SC: O(1)

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



↓ Transpose

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

→ reverse each row

$$O(n^2) + \underbrace{O\left(\frac{n}{2} * n\right)}$$

reverse
every row

Ques)

→ Given a mat[N][N] print all boundaries in clockwise

→ mat[5] [5]

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

output

print 0th row:-

[0 0] → [0 3] : 4

print 4th col

[0 4] → [3 4] : 4

print N-1th row

[4 4] [4 1] : 4

print 0th col

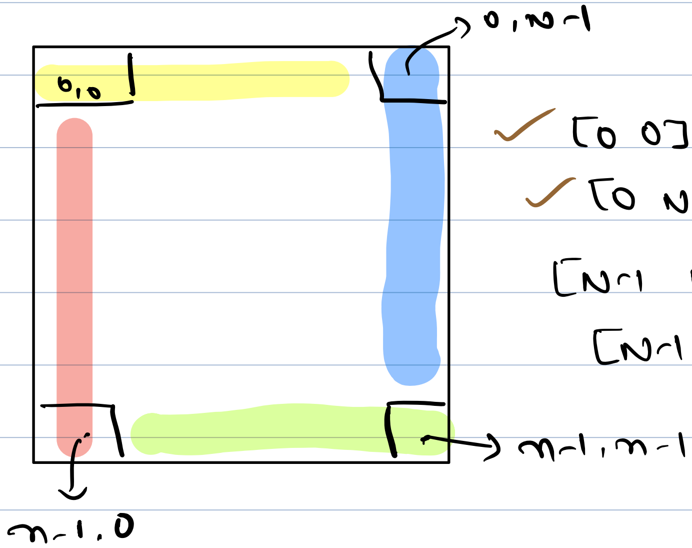
[4 0] . [1 0] : 4

```
i=0, j=0          5
for (k=1; k<~~N~~; k++){     // n-1 times
      print(mat[i][j])
      j++
}
```

k = 1, 2, 3, 4

1 2 3 4

N * N Matrix



✓ [0 0] → [0 n-2] → n-1 ele.

✓ [0 N-1] → [N-2 N-1] → n-1 ele

[N-1 N-1] → [N-1 1] → n-1 ele

[N-1 0] → [1 0] → n-1 ele

```
i=0, j=0
for (k=1; k<N ; k++){      // n-1 times.
    print(mat[i][j])
    j++
}

// i=0   j=n-1

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

// i=n-1 ,   j=n-1

for (k=1; k<N ; k++){      // n-1 times.
    print(mat[i][j])
```

J - -

}

// i = n-1 , J = 0

```
for (k=1; k<N ; k++) {      // n-1 times.
    print (mat[i][J])
    i - -
}
```

// i = 0,   J = 0

Ques) Given matrix [N][N], print spirally.

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

→ mat[4][4]

i = 0, J = 0

while (N>0) {

```
for (k=1; k<N ; k++){      // n-1 times.
    print (mat[i][j] )
    j++
}

// i=0   j= n-1

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

// i = n-1 ,   j = n-1

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

// i= n-1 ,  j=0

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

// i=0,    j=0
i++, j++
n -= 2;
```

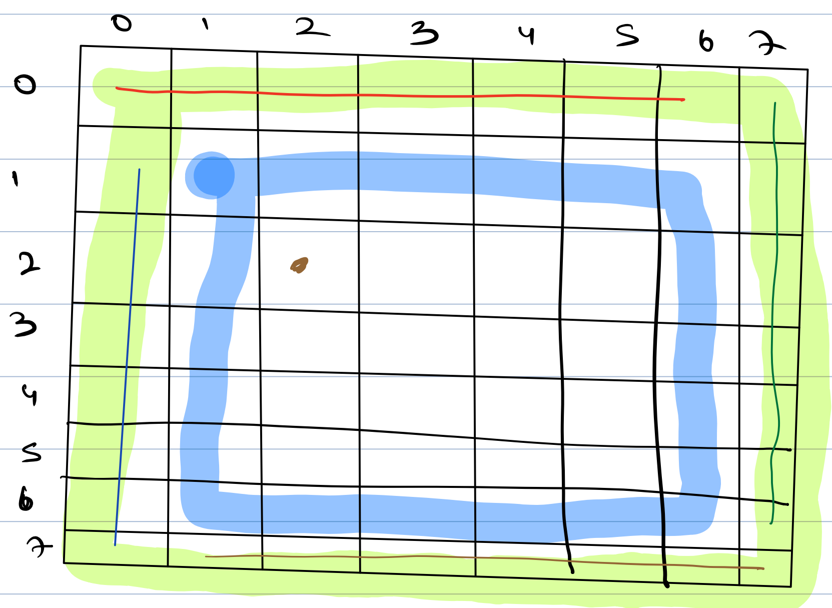
T.C → $O(n^2)$

S.C → $O(1)$.

| 2

if (N == 1) {
    print (the middle guy)
}

mat[8][8] $\underline{n=8}$



1 $\underline{\text{edge case}}$:-

mat [5][5]   n=5

i=0, j=0

|

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

$n = 3$

$i = 1, j = 1$

$i = 2, j = 2$

$n = 1$

$n -= 1$

Modular arithmetic

very big % prime no →

```
int size = A.length;
long totalsum = 0l;
for (int s = 0; s < size; s++) {
long freq = (s + 1) * (size - s);
totalsum += freq * A[s];
}
return totalsum;
```

int

int

int

long

long i = (J) x K

```
for( i=0; i<n; i++) {
        sum = (Arr[i]) % Mod.

}
Ans % Mod
```

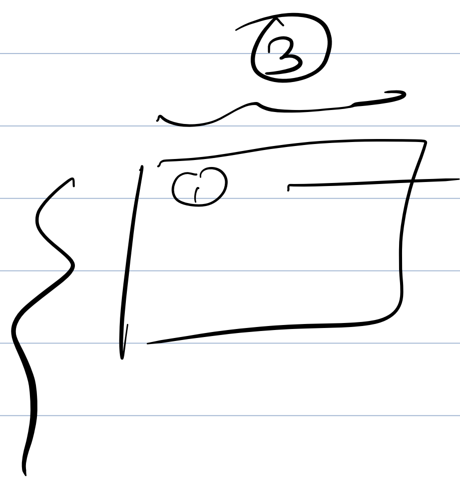
$$(A+B)\%M = A\%M + \underline{B\%M}$$

$$(A + B + C + \ldots Z)\%M$$

$$A\%M + B\%M +$$

$$\underline{1, 2, 3}$$

$$\frac{n(n+1)}{2}$$



Product = $\underline{10} \times \underline{20} \times \underline{30}$

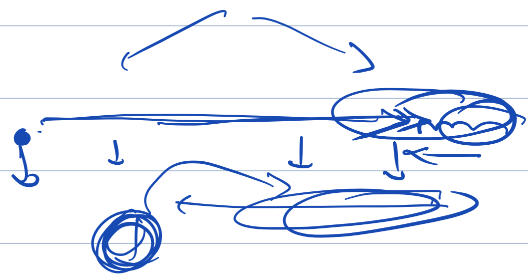
10 20 30 40 50

Suffix Array.

One Prefix query

Buy once and sell once after

cheapest   Expensive



Arrays → 3

Bit map → 2

Maths – 4

Recur → 2

Sorting – 3

B.S → 3

Two pointers → 1

LL → 3

Hashing → 3

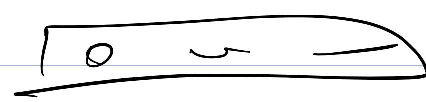
Stacks → 2

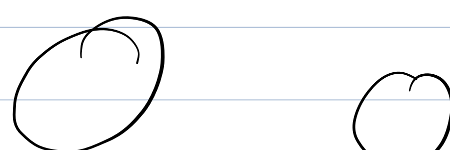
Queue → 2

Trees → 6

Tries - 2

Car [] car --





Greedy → 1 class

String pattern matching → 2 classes.

Graphs - 5

DP - 7

Heaps - 2

BackTracking - 2