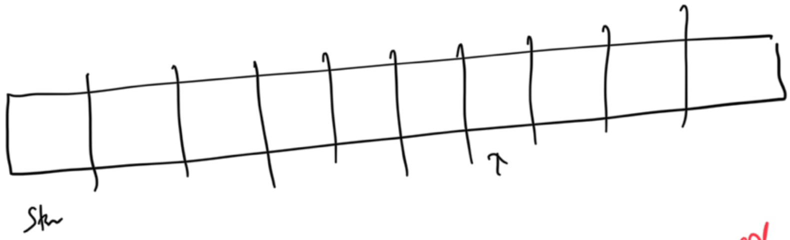
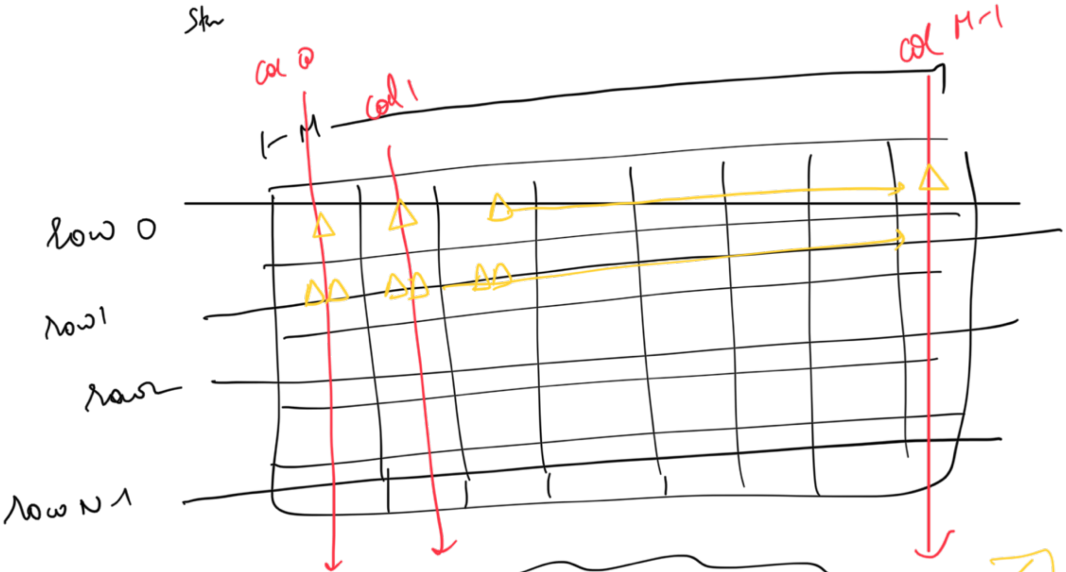
2 D Matrices





2 D Matrix (N, M)

≡

N Arrays of size M combined together

(i, j) ≡ (2, 5)

	0	1	2	3	4	5	6
0	(0,0)	(0,1)	(0,2)	0,3	0,4	0,5	0,6
1	1,0	1,1	1,2	1,3	1,4	1,5	1,6
2	2,0	2,1	2,2	2,3	2,4	(2,5)	2,6
3	3,0	3,1	3,2	3,3	3,4	3,5	3,6
4	4,0	4,1	4,2	4,3	4,4	4,5	4,6
5	5,0	5,1	5,2	5,3	5,4	5,5	5,6

$N \times M$

6×7

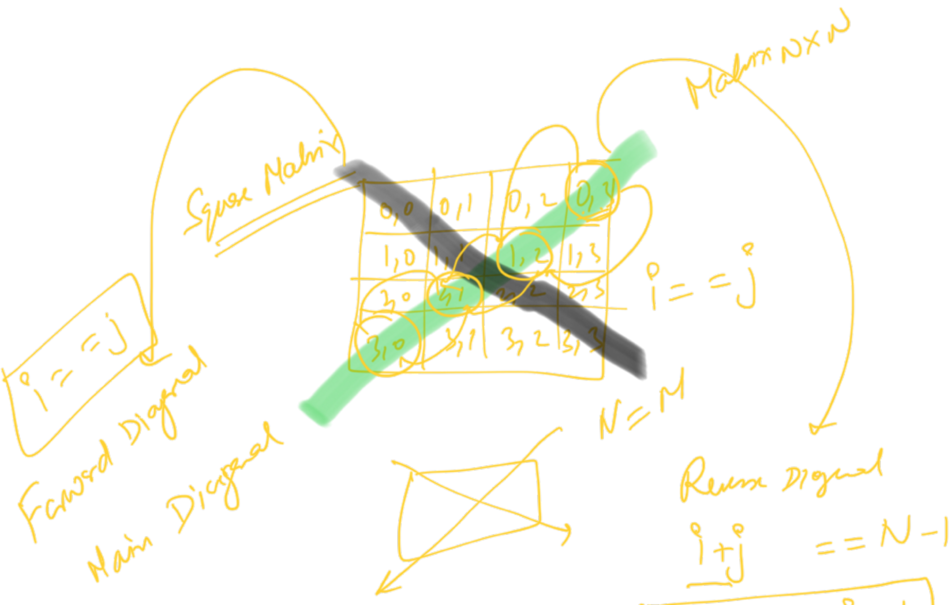
$N = 6$

M = 7

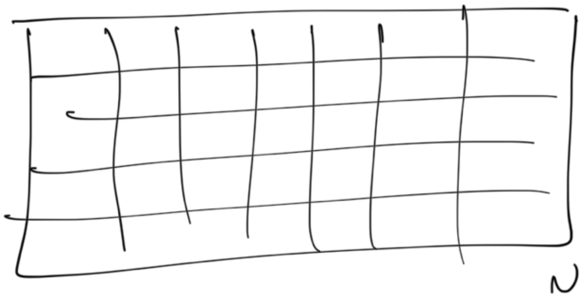
(i, j)

i ≡ rowid

j ≡ colid



Q1


N X M

$O(N)$

Row loop $\equiv$ for (i = 0 , i < N , i++)

{

Col loop $\equiv$ for (j = 0 , j < M , j++)

$O(M)$

{ print (A[i][j])

}

}

Row Major format

$T.C = O(N \times M)$

compiler $S.C = O(1)$

Col MAJOR FORMAT

N X M

Col loops $\equiv$ for (j = 0 , j < M , j++)

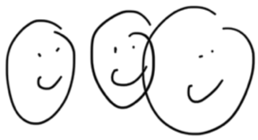
{

Row loop = for(i=0, i<N, i++)

{

print A[i][j]

}



}

Q

$$A = \begin{array}{ccc} \text{col0} & \text{col1} & \text{col2} \\ \begin{bmatrix} 3 & 1 & 6 \\ 8 & 2 & 5 \\ 0 & -1 & 3 \end{bmatrix} \end{array}$$

$$\begin{array}{ccc} 11 & 2 & 14 \end{array}$$

```
for( j=0 , j<M, j++)
{
    sum=0
    for( i=0, i<N, i++)
    {
        sum = sum+ A[i][j]
    }
}
```

T.C = O(N×M)

S.C = O(1)

= O(N×M)

awp input S.C print (sum)

}

Q M_1 $(N \times M)$ + M_2 $(N \times M)$

Answer

$$\begin{bmatrix} 2 & 2 & 8 \\ 1 & 6 & 9 \end{bmatrix}_{2\times3} + \begin{bmatrix} 0 & 6 & 10 \\ 1 & 2 & 4 \end{bmatrix}_{2\times3}$$

$$= \begin{bmatrix} 3 & 8 & 18 \\ 2 & 8 & 13 \end{bmatrix}_{2\times3}$$

2D Matrices

(A) Traverse every single element of the matrix

(Q) Which format of traversal (row major / col major) will be helpful?

*
*
*
*

```
SumTwoMatrices( A[N][M] , B[N][M] )
{        answer[N][M];

    for(i=0, i<N, i++)
    {
         for(j=0, j<M, j++)
         {
              answer[i][j]
                 =
              A[i][j] + B[i][j];
         }
    }
```

T.C = O(N×M)

S.C = O(N×M)
ans

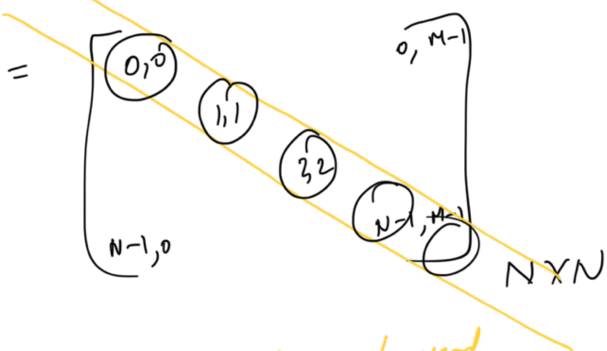
input S.C = O(N×M)

Quiz

$[\quad]_{N \times M}$ $[\quad]_{M \times N}$

$[\quad]_{2 \times 3}$ $[\quad]_{3 \times 2}$

Q square A =



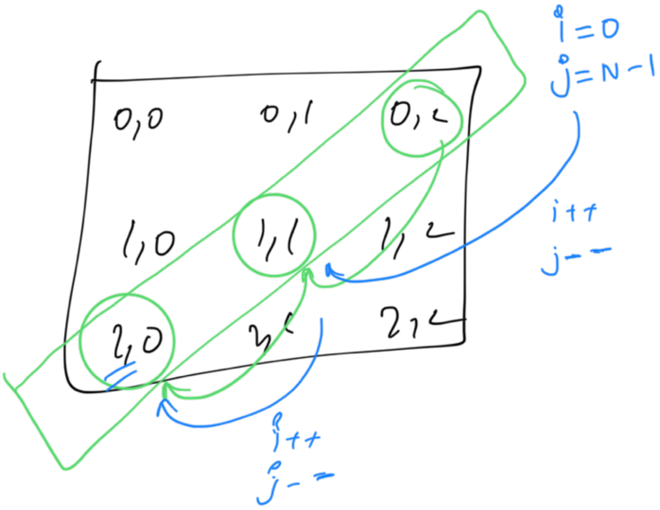
Print elements of the forward.

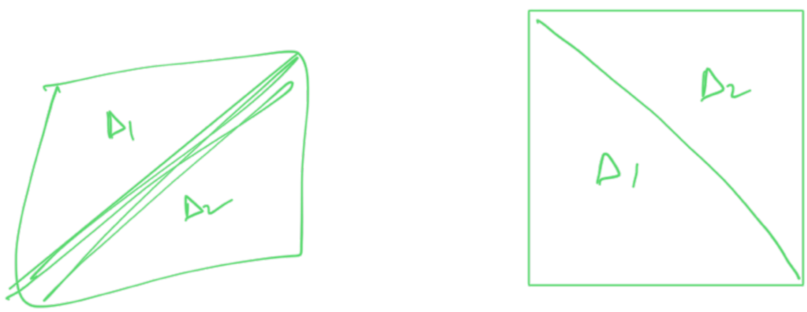
```
for (i = 0, i < N, i++)
    print(A[i][i])
```

$T.C = O(N)$ ✓

aux $S.C = O(1)$

input $S.C = O(N \times M)$





I

```
for (i=0, i<N, i++)
    print(A[i][N-i-1])
```

A[0][2]
A[1][1]
A[2][0]

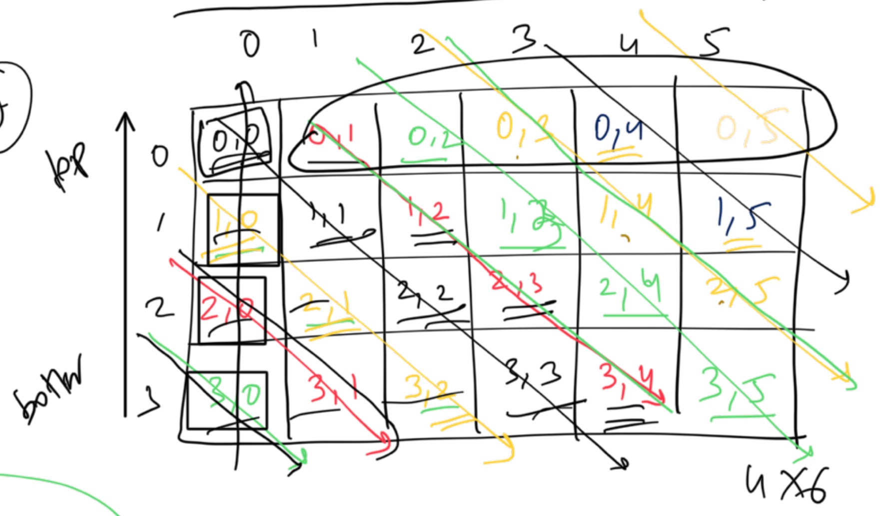
II

```
i = 0
j = N-1
while (i < N)
{
        print (A[i][j])
        i++
        j--
}
```

T.C = O(N)
aux S.C = O(1)

left to right

0



Invariant

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

$\epsilon_k \to \epsilon_{k+1}$

$i \to i+1$

$j \to j+1$

D_1

D_2 0,1 1,2 2,3 3,4

D_3

Invariant

In a changing landscape
what property remains constant

Q) Print all elements in the forward diagonal direction one by one

O(NxM)

```
printDiagonalElements( A[N][M], startI
                                startJ )
{
    i = startI
    j = startJ
    while( i<N && j<M)
    {
        print (A[i][j])
```

i++

j++

}

}

We just need to figure out which all starting points are there

and then just call this function for every starting point

Col 0

Top

Bot

```
// Loop → Col 0 from bottom to Top
for(i = N-1, i ≥ 0, i--) {
    startI = i
    startJ = 0
    printdiagonalElements(A[N][M], startI, startJ)
}
```

// Loop 2 → row 0 from L to R

for(j = 1 , j < M , j++)

{

startI = 0

startJ = j

printDiagonalElements (A[i][j] , startI, startJ)

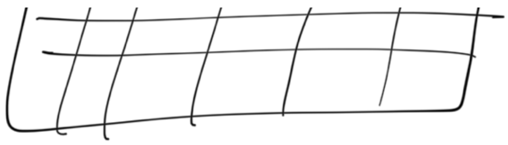
}

T.C = O(N × M)

aux S.C = O(1)

Q

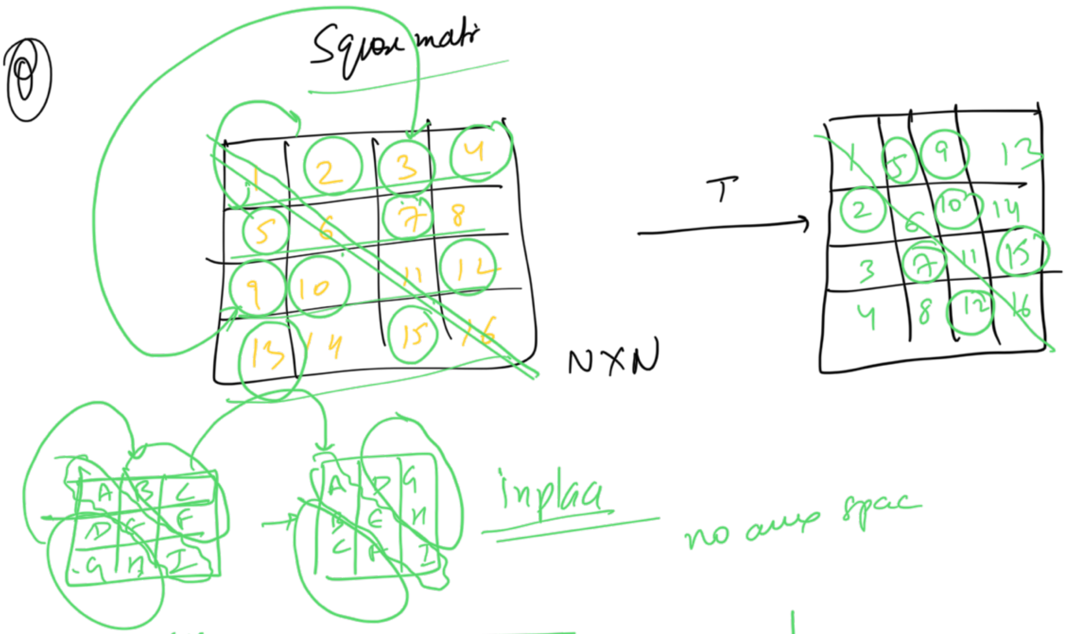
rc N × M



(A) Find the invariant

(B) Invariant for the starting pts

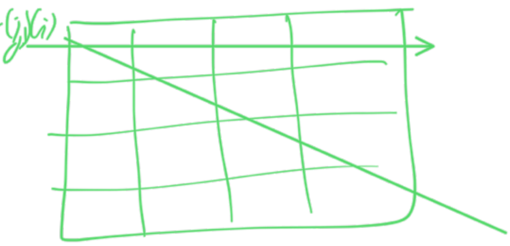
(Q) Square matrix



inplace

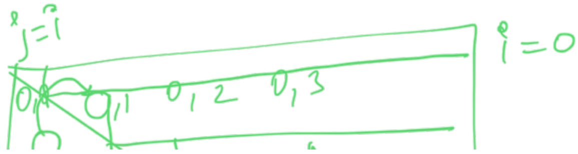
no aux space

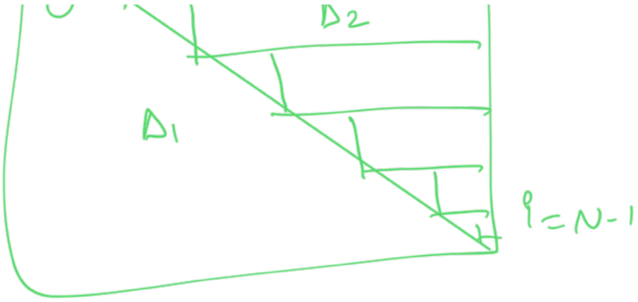
A(i)(j) ⟷ A(j)(i)



MxN

i, j → j, i



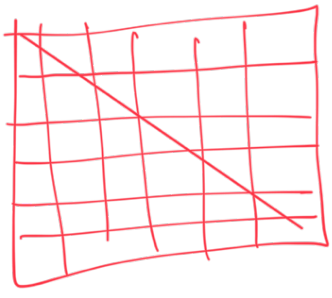


swap (A(i)(j) , A(j)(i))

upper Δ

```
for( i=0 , i<N; i++)
{
    for( j=i+1, j<M, j++)
    {
        Swap ( A(i)(j) , A(j)(i) )
    }
}
```

$O(N^2)$

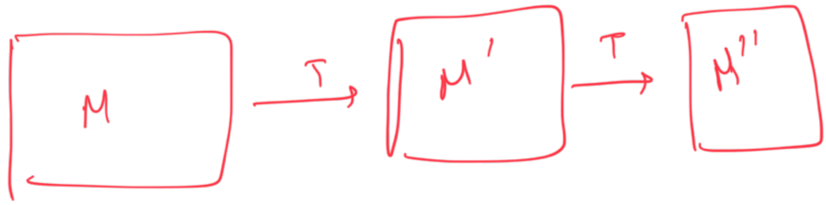


i=0 to N

j=0 to M

swap every elem twice

no change matrix

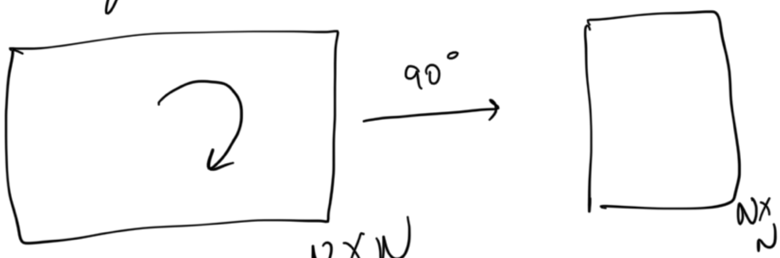


M == M'

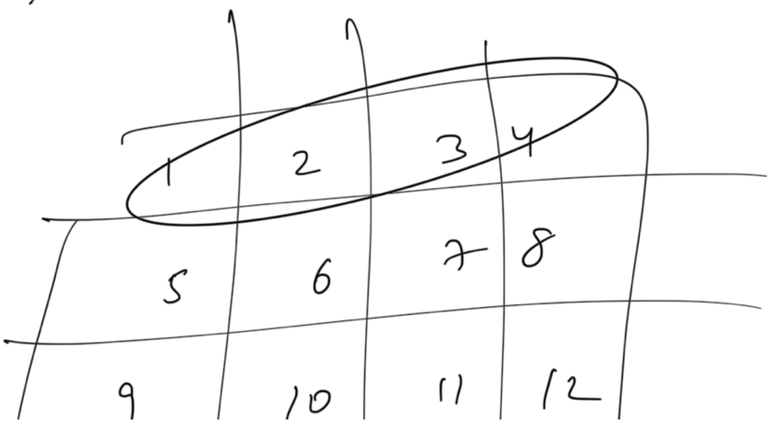
0

Apple meta

Square M



'inplace'



$$\begin{array}{|c|c|c|c|} \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

↓ 90° ↻

$$\begin{bmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{bmatrix}$$

A Rows ⟷ Cols Both

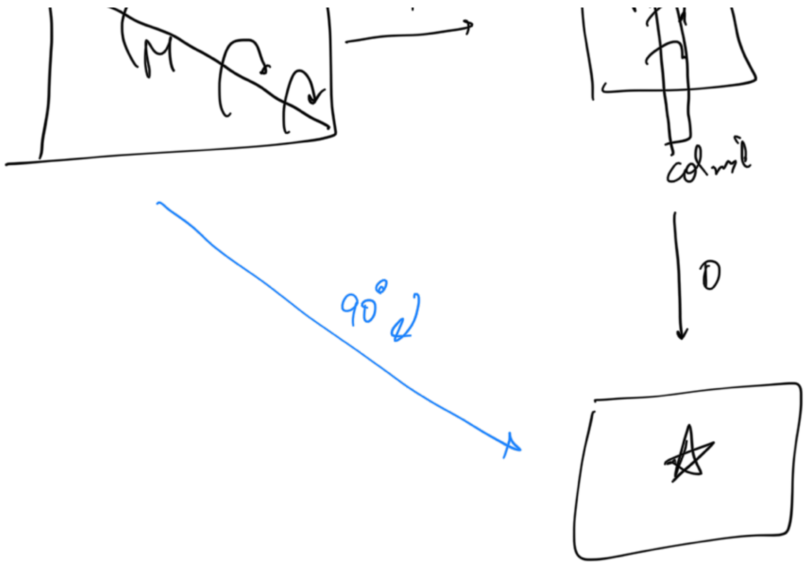
was being performed both in Transpose and in 90° ↻

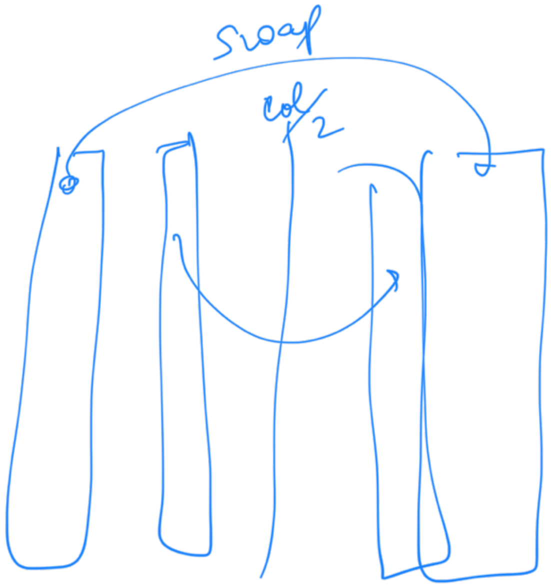
B Transpose

row0 ⟷ col 0
row i ⟷ col i

B 90° ↻

row0 ⟷ col N-1
row i ⟷ col N-1-i


column
0
90°


swap
col/2