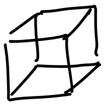
2D Matrices

Jul 7, 2023

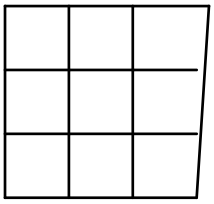
AGENDA

- Intro to 2D Matrices
- Row wise / Col wise sum
- Transpose of a matrix
- Rotate matrix
- Print diagonals of a matrix

2-D

•	Point	0-dimensional	Single integer element
—	Line	1-dimension	Array
	Square/Rectangle	2 dimension	2D-Array / Matrix
	Cube	3 dimension	3D-Array.

Matrix

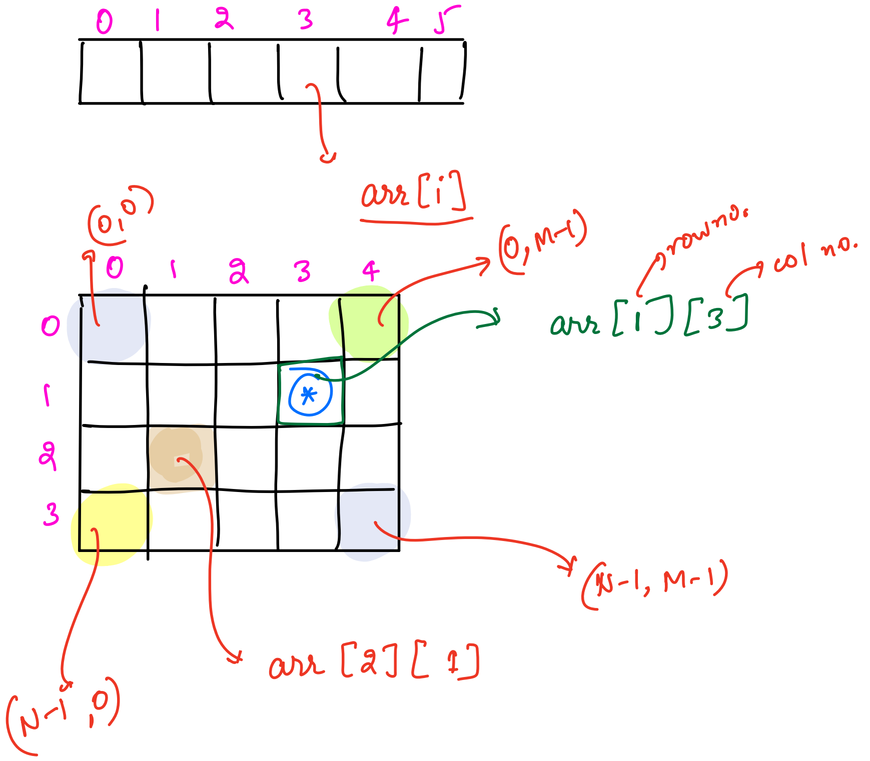


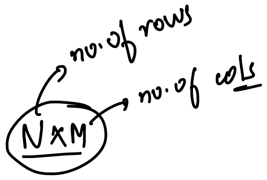
Horizontal

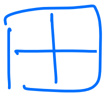
No. of rows → 3

No. of columns → 3

Vertical







* Access ith row of a matrix

↓ arr[i]

$$
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 1 & 2 & 4 & 6 & 2 & 3 \\
1 & 8 & 0 & 3 & 5 & 0 & 0 \\
2 & 7 & 7 & 2 & 7 & 6 & 7 \\
3 & 5 & 0 & 1 & 2 & 9 & 8
\end{array}
$$

$\boxed{\text{arr}}$

$\boxed{\text{arr[0]}}$ - $\{\overset{0}{1}, \overset{1}{2}, \overset{2}{4}, \overset{3}{6}, \overset{4}{2}, \overset{5}{3}\}$

arr[0][0] = 1

arr[3] = $\{\overset{0}{5}, \overset{1}{0}, \boxed{\overset{2}{1}}, 2, 9, 8\}$

arr[3][2] = 1

* Access jth column of a matrix

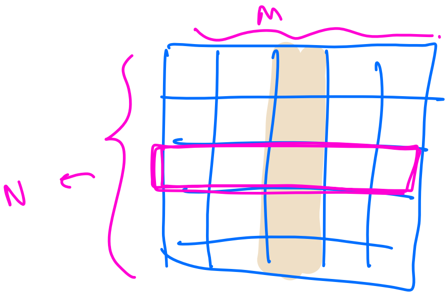
arr[j] → XX it will give jth row

arr[0][j] → X X element at 0,j

arr[][j] → X X incorrect syntax

You cannot access jth column directly.

Q. **Print ith row**   N x M



```
for (int j=0; j< M ;j++)
{
        print ( arr[i][j])
}
```

fixed.

Q. **Print ith column**

```
for( int j=0; j< N ;j++)
{
        print( arr [j][i])
}
```

fixed.

Q. **Print row-wise sum of a matrix**   N x M

```
for(int i=0 ; i< N ; i++)
{
        sum=0
        for(int j=0; j< M ;j++)
        {
                sum += arr [i][j]
        }
        print(sum)
}
```

$$\begin{array}{c|cccccc|l} i \backslash j & 0 & 1 & 2 & 3 & 4 & 5 & \\ \hline 0 & 1 & 2 & 4 & 6 & 2 & 3 & : 18 \\ 1 & 8 & 0 & 3 & 5 & 0 & 0 & : 16 \\ 2 & 7 & 7 & 2 & 7 & 6 & 7 & : 36 \\ 3 & 5 & 0 & 1 & 2 & 9 & 8 & : 25 \end{array}$$

T.C. → $O(N*M)$

S.C. → $O(1)$

Q. Print col-wise sum of a matrix

```
for(int j=0; j<M ;j++)
{
        sum=0
    for(int i=0; i<N; i++)
    {
            sum += arr[i][j]
    }
    print(sum)
}
```

$$\begin{array}{c|cccccc}
 & & & & j & & \\
i & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 1 & 2 & 4 & 6 & 2 & 3 \\
1 & 8 & 0 & 3 & 5 & 0 & 0 \\
2 & 7 & 7 & 2 & 7 & 6 & 7 \\
3 & 5 & 0 & 1 & 2 & 9 & 8 \\
\hline
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 & 21 & 9 & 10 & 20 & 17 & 18
\end{array}$$

H.W : Try to iterate row-wise and print sum col-wise

$$\begin{array}{|c|c|c|c|}
\hline
2 & 3 & 1 & 0 \\
\hline
0 & 0 & 7 & 2 \\
\hline
8 & 1 & 2 & 1 \\
\hline
\end{array}$$

$$\begin{array}{cccc}
[\cancel{0} & \cancel{0} & \cancel{0} & \cancel{0}] \\
\cancel{2} & \cancel{3} & \cancel{1} & \cancel{0} \\
\cancel{2} & \cancel{3} & \cancel{8} & \cancel{2} \\
10 & 4 & 10 & 3
\end{array}$$

10 , 4 , 10, 3

Q: Transpose of a square matrix (N=M)

$$\underset{\text{inp}}{\begin{bmatrix}2&3&7\\1&2&4\\0&1&5\end{bmatrix}} \longrightarrow \underset{\text{transpose}}{\begin{bmatrix}2&1&0\\3&2&1\\7&4&5\end{bmatrix}}$$

(ith row becomes ith column.)

Q: Transpose a matrix NXN in-place. (Do not use extra space.)

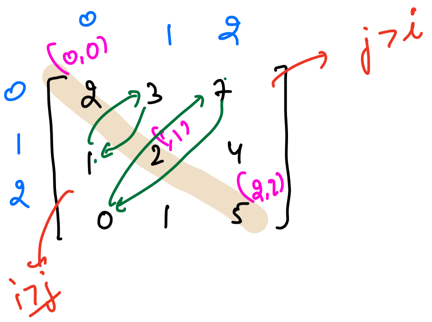
$$\begin{array}{c|c|c|c|c|} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 4 & 7 \\ \hline 1 & 8 & 0 & 3 & 5 \\ \hline 2 & 7 & 7 & 2 & 7 \\ \hline 3 & 5 & 0 & 1 & 2 \\ \hline \end{array} \longrightarrow \begin{array}{cccc} 1 & 8 & 7 & 5 \\ 2 & 0 & 7 & 0 \\ 4 & 3 & 2 & 1 \\ 7 & 5 & 7 & 2 \end{array}$$

(2,0)

(2,0) element → (0,2)

(2,3)

$(i,j)^{th}$ element in input → $(j,i)^{th}$ element in transpose.



$$\begin{array}{cccccc} i & & & & & j \\ 3, & 7, & 2, & 0, & 5, & 6 \end{array}$$

```
for(int i=0; i<N; i++)
{
        for(int j=0; j<N; j++)
        {
                // swap (i,j) with (j,i)
                int temp = arr[i][j]
                arr[i][j]= arr[j][i]
                arr[j][i]= temp

        }
}
```

wrong code.

It will give back the same matrix.

Only iterate in the top right or bottom left.

```
for(int i=0; i<N; i++)
{
        for(int j=0; j<i; j++)
        {
                // swap (i,j) with (j,i)
                int temp = arr[i][j]
                arr[i][j]= arr[j][i]
                arr[j][i]= temp
        }
}
```

bottom left

or for(int j=i+1;j<n;j++)

top right.

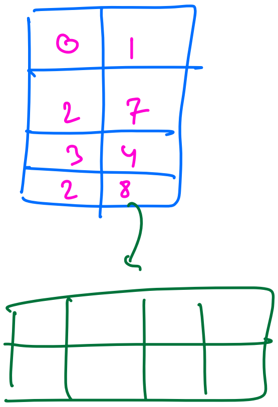
T.C. → $O(N^2)$

S.C. → $O(1)$

* Transpose a N*M matrix.

{

will have to use extra space.



0	1
2	7
3	4
2	8

```
for(int i= 0 ; i< m ; i++)
    for(int j=0 ; j<N ; j++)
        new_arr[i][j]= arr[j][i]
```

Q: Given a square matrix, rotate it by 90° in clockwise direction. (without extra-space)

$$\begin{bmatrix} 2 & 3 & 7 \\ 1 & 2 & 4 \\ 0 & 1 & 5 \end{bmatrix} \xrightarrow{\circlearrowright} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 5 & 4 & 7 \end{bmatrix}$$

* Rows become columns.

* A certain row becomes a certain column.

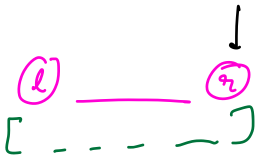
$$\begin{bmatrix} 2 & 3 & 7 \\ 1 & 2 & 4 \\ 0 & 1 & 5 \end{bmatrix} \xrightarrow[\text{Transpose}]{} \begin{bmatrix} 2 & 1 & 0 \\ 3 & 2 & 1 \\ 7 & 4 & 5 \end{bmatrix} \xrightarrow[\text{Reverse each row one by one.}]{}$$

Rotate by 90° = Transpose + Reverse each row

Code

```
// arr[][] is Input.
transpose (arr);        ← Use previous code.

for(int i=0 ;i< N ;i++)
{
        //Reverse arr[i] ;
        l = 0
        r = N-1
        while (l<r)
        {
                temp= arr[i][l]
                arr[i][l] = arr[i][r]
                arr[i][r] = temp

                l++
                r--
        }
}
```

$$\begin{array}{ccccc} & l & & & r \\ & 7\,\cancel{3} & 2\,\cancel{1} & \cancel{2}\,1 & \cancel{7}\,3 \\ & 2 & 8 & 0 & 1 \\ i & 3 & 0 & 1 & 4 \\ & 1 & 6 & 2 & 9 \end{array}$$



0, N-1

1, N-2

2, N-3

⋮

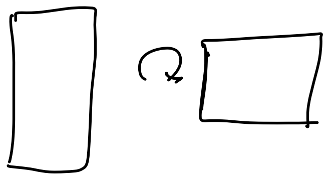
N/2

T.C. → $O(N^2)$

S.C. → $O(1)$

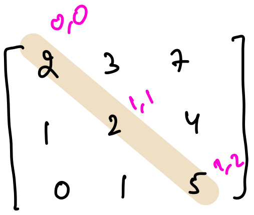
* Rotate a matrix (NXM)

↳ Need extra space

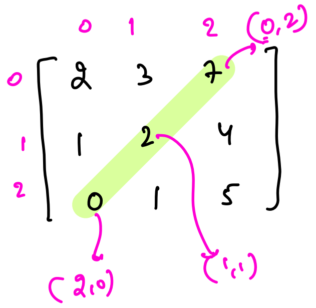


Break till 8:28 AM

Q: Print the diagonals in a square matrix.



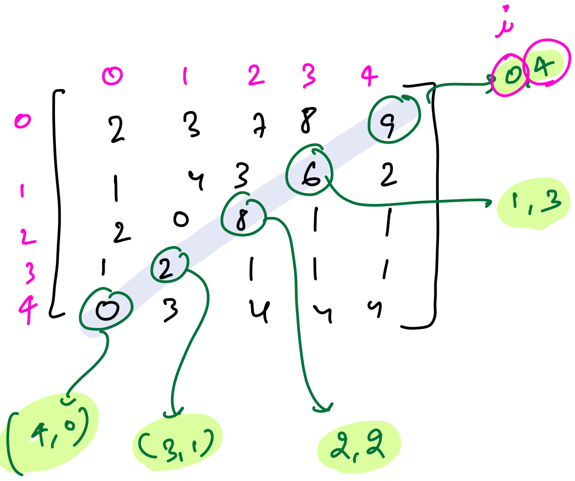
```
for(int i=0; i<n;i++)
{     print (arr[i][i])
}
```



$(i, n-i-1)$

```
for(int i=0 ; i< n;i++)
{
        print(arr[i][n-i-1])
}
```

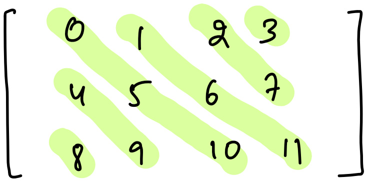
[7, 2, 0]



Total = n-1

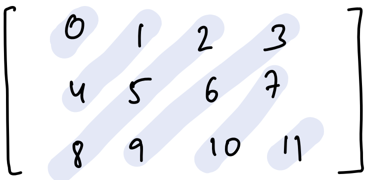
$(i, n-1-i)$

Diagonals in a rectangular matrix



$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix}$$

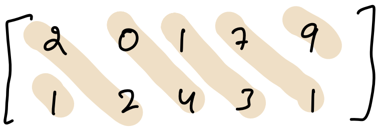
6 diagonals.
(Left-to-right)



$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix}$$

6 Anti-diagonals.
(right-to-left)

<u>N=2, M=5</u>

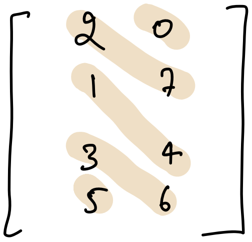


$$\begin{bmatrix} 2 & 0 & 1 & 7 & 9 \\ 1 & 2 & 4 & 3 & 1 \end{bmatrix}$$

No. of diagonals ? <u>6</u>

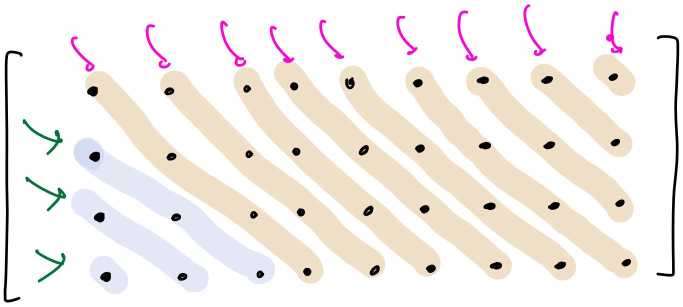
NxM matrix, how many diagonals are there ?

N = 4, M=2



$$\begin{bmatrix} 2 & 0 \\ 1 & 7 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

No. of diagonals ? <u>5</u>



M + N − 1
diagonals

Q: Given a matrix of size N×M, print all the diagonals.

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix}$$

Output:

```
8
4  9
0  5  10
1  6  11
2  7
3
```

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

1 : [2,0]
2 : [1,0] , [2,1]
3 : (0,0) , (1,1) , (2,2)
4 : (0,1) , (1,2) , (2,3)
5 : (0,2) , (1,3)
6 : (0,3)

Observations

1. In a diagonal, if current cell is (i,j) , next cell is (i+1, j+1).

2. Keep on printing a diagonal until you are outside the boundary.
   (Only print as long as . $i < n$ && $j < m$)

3. Start point of 1st diagonal is : (N-1, 0) [Bottom left.]

4. To change diagonal, go up → up → up → until you reach (0,0). → go right → right → right.

Code.

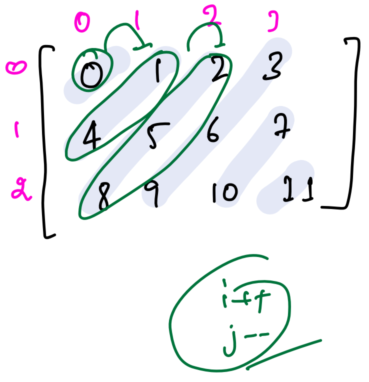
```
r = N-1,
c = 0
while ( c < M ) {

        //(r,c) is the start point of a diagonal.
        i = r
        j = c
        while ( i<n && j<m)
        {
              print( arr[i][j])
              i++
              j++

        }
        print("\n");
        // Change diagonal
        if (r != 0)
        {      r-- ;        (go top)

        } else
        {
              c++;          (go right)

        }
}
```

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & \overset{i,j}{1} & 2 & 3 \\ 1 & 4 & 5 & \overset{i,j}{6} & \overset{}{7} \\ 2 & 8 & 9 & 10 & \overset{i\,j}{11} \end{array} \quad i\,j$$

1 6 11

T.C. → O(N * M)

H.W ⁚)



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

i++
j--

(0,0)

Lecture Day

↓

Non-Lecture Day (Assignment ✓)

Homeworks ⁚)

Sunday → focus on homework problems ⁚