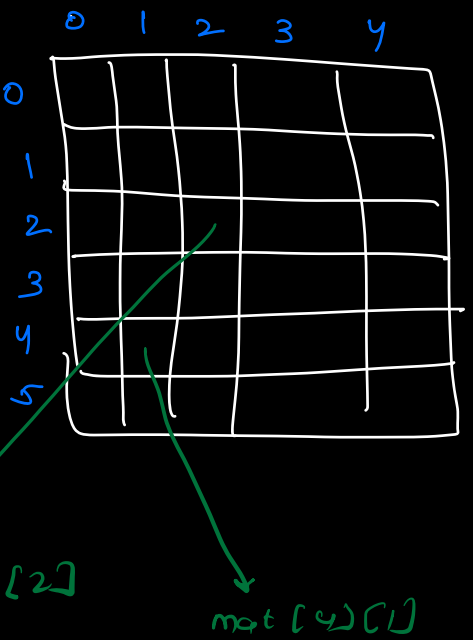
```
int arr[];

int mat[][] = new int[6][5];
```

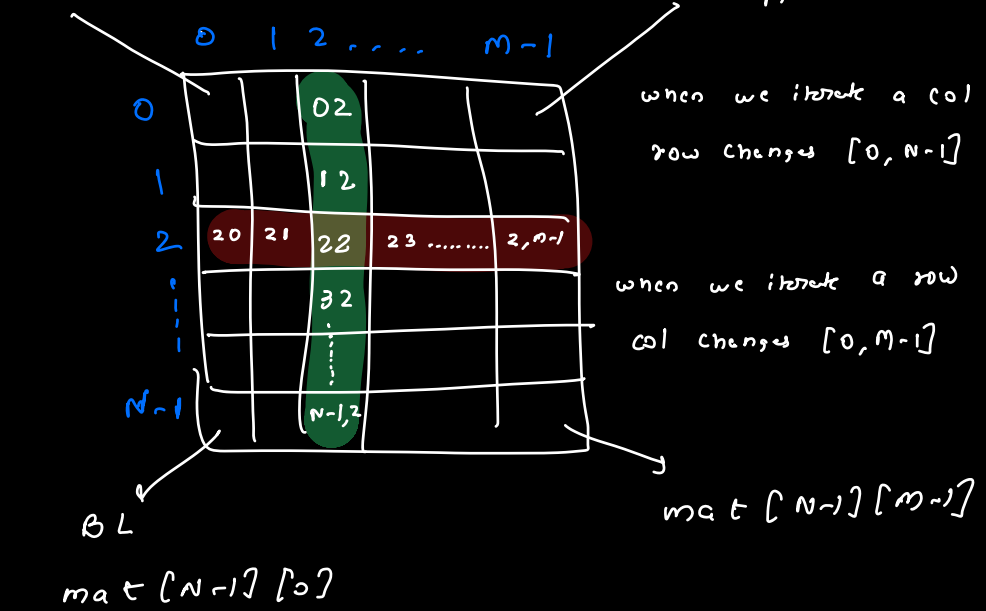
R C

rows

columns.

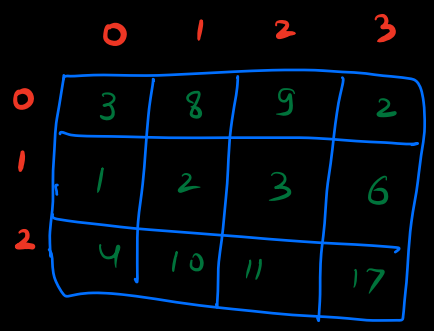


mat[N][M]



Q. Given mat[N][M], print row-wise sum.

mat[3][4]



Output

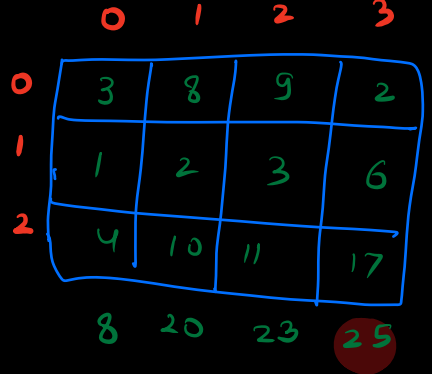
22
12
42

```
for(i=0 ; i<N; i++)
{
    // ith row
    sum = 0
    for(j=0; j<m; j++)
    {
        sum = sum + mat[i][j]
    }
    print(sum)
}
```

TC : O(N×m)
SC : O(1)

Q. Given mat[N][m], find max col-sum.

mat[3][4]



ans = 25

```
maxi = INT_MIN
for(j=0; j<m; j++)
{
    // jth column
    sum=0
    for(i=0; i<N; i++)
    {
        sum = sum + mat[i][j]
    }
    if(sum > maxi)      maxi = sum
    maxi = math.max(maxi, sum)
}
return max
```

OR

Tc : O(N×M)

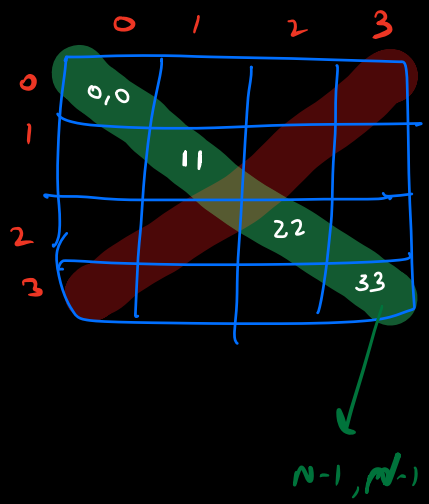
Sc : O(1)

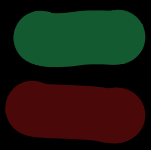
```
void fun(int mat[][])
{
    int N = mat.length
    int m = mat[0].length
}
```

Q. Given a mat[N][N].

print both diagonal (TL → BR , TR → BL)

mat[4][4]

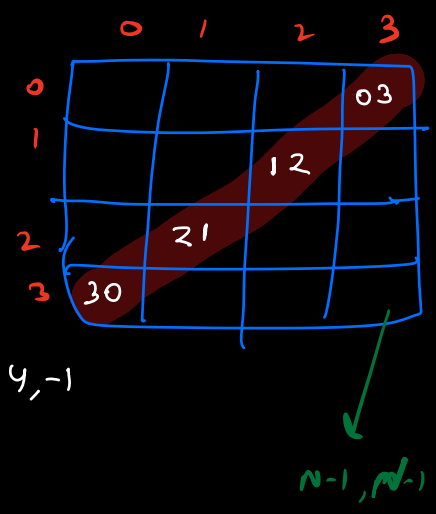




only one is enough

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

TC: O(N)
SC: O(1)



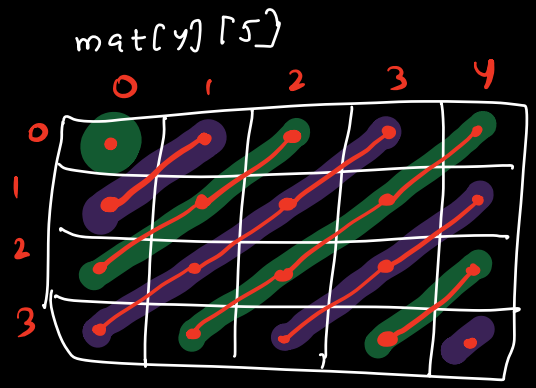
Only one is sufficient

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

TC: O(N)
SC: O(1)

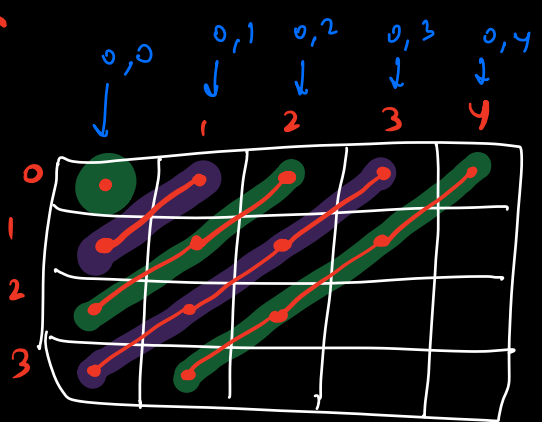
Total TC: O(N)
SC: O(1)

Q. Given mat[N,M], print all the diagonals going from (T-B) (R-L) for one diagonal.



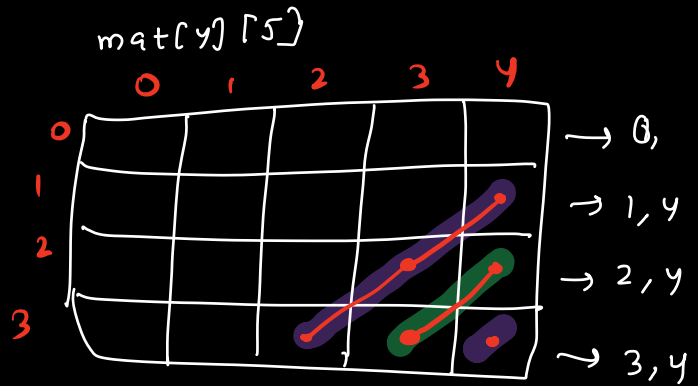
Observation → Starting points lie on first row or last column.

```
// 0th row starting diagonals.
for ( j = 0 ; j < m ; j++ )
{
    // print the diagonal starting at 0th row, jth col
    x = 0, y = j
    while ( x < N && y >= 0 )
    {
        print ( mat[x][y] )
        x++ , y--
    }
}
```



```
for ( i = 1 ; i < N ; i++)
{
    // print diagonal starting at i^th, m-1

    x = i , y = m-1
    while( x<N && y≥0 )
    {
        print ( mat[x][y] )
        x++ , y--
    }
}
```

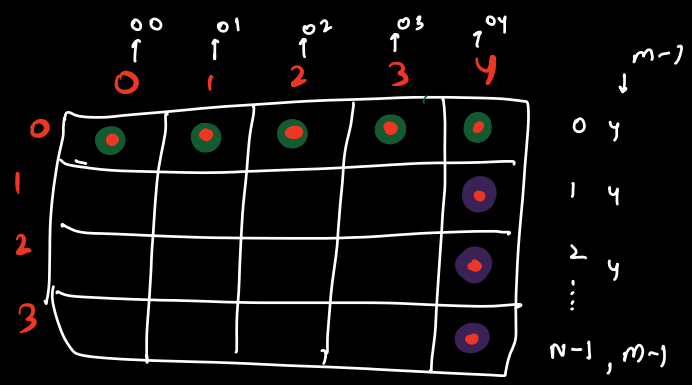


TC : O(N x M)

SC : O(1)

Rewriting above solution

```
void printdig (int i, int j)
{
    x = i, y = j
    while( x < N && y >= 0 )
    {
        print ( mat[x][y] )
        x++, y--
    }
}
```



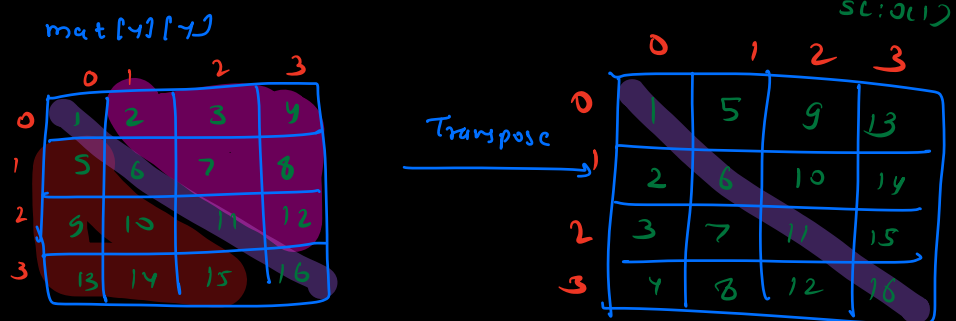

```
for( j=0; j<m; j++)
{
    printdia(0, j)
}

for( i=1; i<N; i++)
{
    printdia( i, m-1)
}
```

TC: O(N+m)
SC: O(1)

Q. Given a mat[N][N], find the transpose.

SC: O(1)



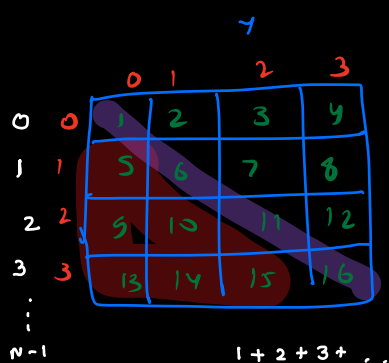
mat[i][0] ⟷ mat[0][i]

3,0 ⟶ 0 3

3 1 ⟶ 1,3

mat[i][j] ⟷ mat[j][i]

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



1+2+3+ ... N-1

$\frac{(N)(N-1)}{2} = O(N^2)$

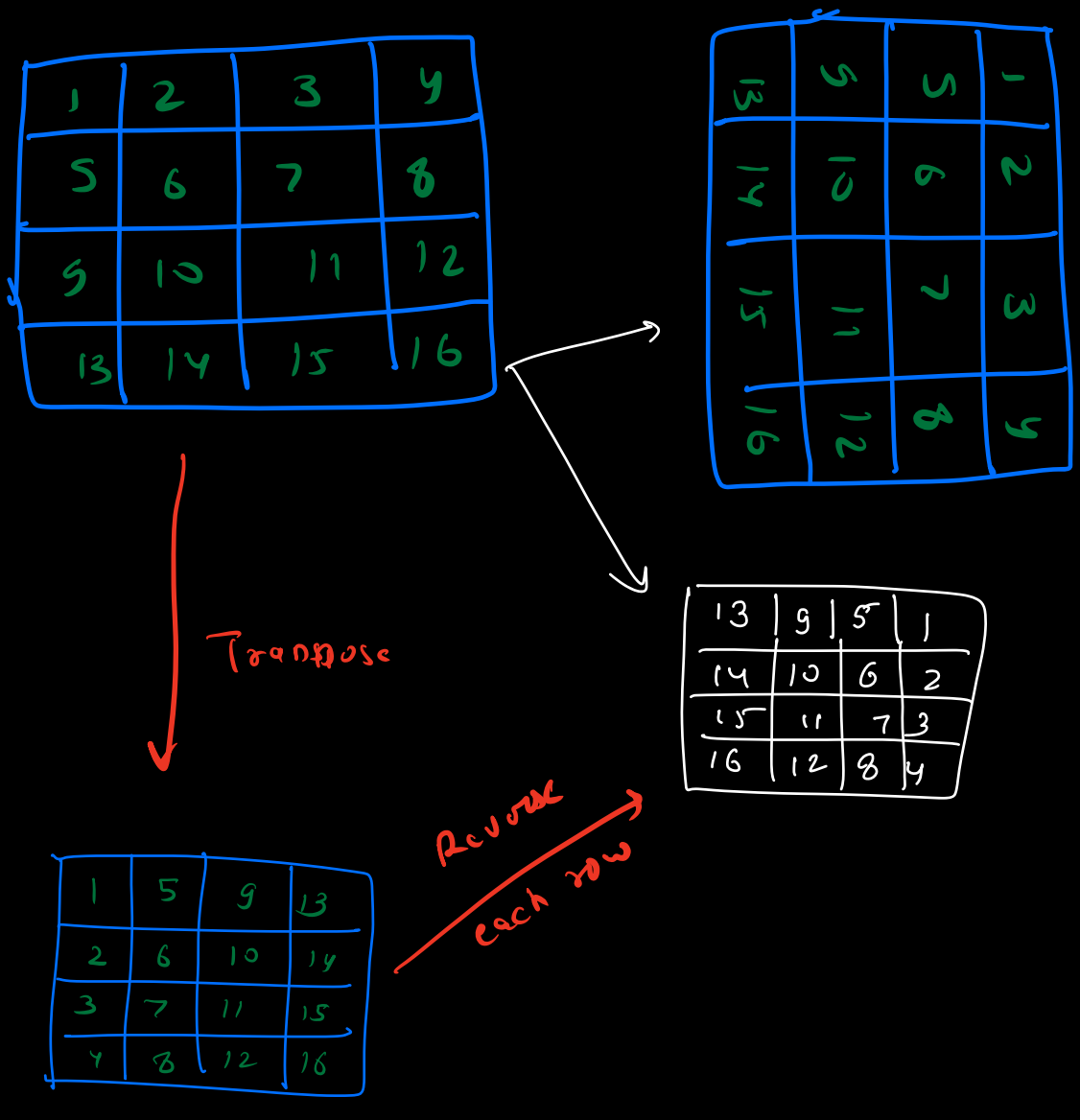
TC: $O(N^2)$

SC: $O(1)$

```
void swap( int a, int b)
{
    c = a
    a = b
    b = c
}
```

Q. Given a square mat[N][N]. Rotate it by 90° clockwise.

SC : O(1)



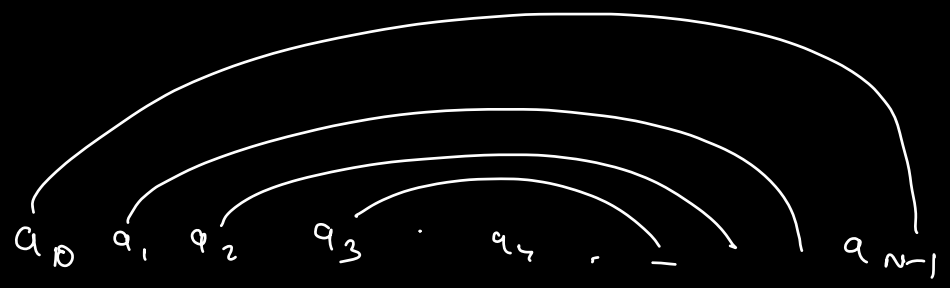
Rotate 90° = Transpose + Reverse each row

TC : $O(N^2)$ TC : $O(N^2)$

TC : O(1) SC : O(1)

= TC : O(N²)

SC : O(1)




```
i=0, j=N-1
while (i<j)
{
    swap(arr[i], arr[j])
    i++, j--
}
```

arr[2]

5
4

arr[N-1]

4
5