Today's agenda

- Transpose
- Rotate matrix by 90 degrees
- Range sum query 2D
- Sum of all submatrix sum

Q) Transpose of a matrix

↳ Given a mat[N][N], calculate transpose of mat[][]
(convert every col into corresponding row.)

Note: you can't use a new matrix.

Ex:

	0	1	2	3
0	10	20	30	40
1	50	60	70	80
2	90	100	110	120
3	130	140	150	160

→

	0	1	2	3
0	10	50	90	130
1	20	60	100	140
2	30	70	110	150
3	40	80	120	160

(0,1) (1,0)
(1,0) (0,1)

(1,2) (2,1)
(2,1) (1,2)

$(i,j) \rightleftarrows (j,i)$

↳ Transpose is equivalent to swapping (i,j) & (j,i).

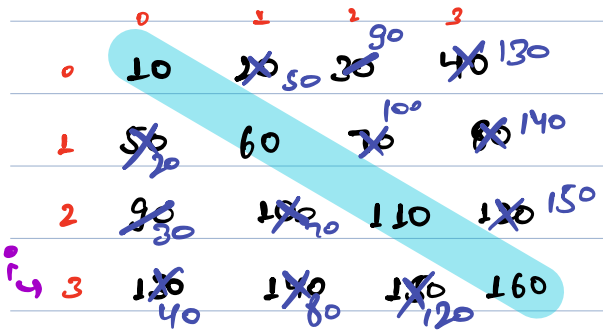
//Pseudo code

```
void transpose (int arr[N][N]){
    for (int i=1; 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;
        }
    }
}
```

T.C: $O(n^2)$

S.C: $O(1)$

//Tracing




```
for(int i=1; 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;
    }
}
```

i=1 → j=0 Swap (1,0) with (0,1)
↳ j=1 → exit

i=2 → j=0 Swap (2,0) with (0,2)
↳ j=1 Swap (2,1) with (1,2)
↳ j=2 → exit

i=3 → j=0 Swap (3,0) with (0,3)
↳ j=1 Swap (3,1) with (1,3)
↳ j=2 Swap (3,2) with (2,3)
↳ j=3 → exit

i=4 → exit

// what if matrix is rectangle?

mat

	0	1	2
0	1	2	3
1	4	5	6

2×3

transpose

	0	1
0	1	4
1	2	5
2	3	6

3×2

↳ matrix changed → It is not possible to take transpose within the same matrix.

↳ transpose[i][j] = mat[j][i];

Q) Rotate the matrix → {Google, Amazon}

↳ Rotate the given mat[N][N] clockwise by 90°

Ex:

	0	1	2	3
0	10	20	30	40
1	50	60	70	80
2	90	100	110	120
3	130	140	150	160

→

	0	1	2	3
0	130	90	50	10
1	140	100	60	20
2	150	110	70	30
3	160	120	80	40

Step1: Transpose of the matrix

	0	1	2	3
0	10	50	90	130
1	20	60	100	140
2	30	70	110	150
3	40	80	120	160

Step2: reverse every row →

	0	1	2	3
0	130	90	50	10
1	140	100	60	20
2	150	110	70	30
3	160	120	80	40

// Pseudo code

```
void Rotate 90° clockwise (int arr[N][N]){
    //Step1
    transpose (arr);

    //Step2
    for (int i=0; i<N; i++) {
        int[] temp = arr[i];
        reverse (temp); → {TODO}
    }
}
```

T.C: $O(N^2)$

S.C: $O(1)$

Q) Rotate the given mat[N][N] anti-clockwise by 90°

Ex:

	0	1	2	3
0	10	20	30	40
1	50	60	70	80
2	90	100	110	120
3	130	140	150	160

Step1: transpose

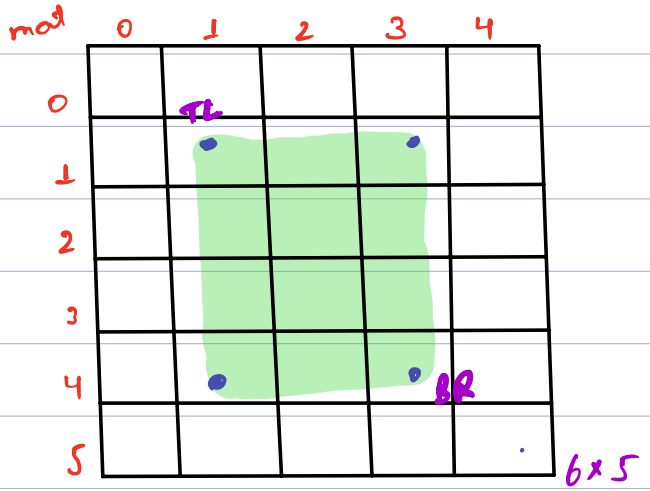
	0	1	2	3
0	10	50	90	130
1	20	60	100	140
2	30	70	110	150
3	40	80	120	160

Step2: reverse Column

	0	1	2	3
0	40	80	120	160
1	30	70	110	150
2	20	60	100	140
3	10	50	90	130

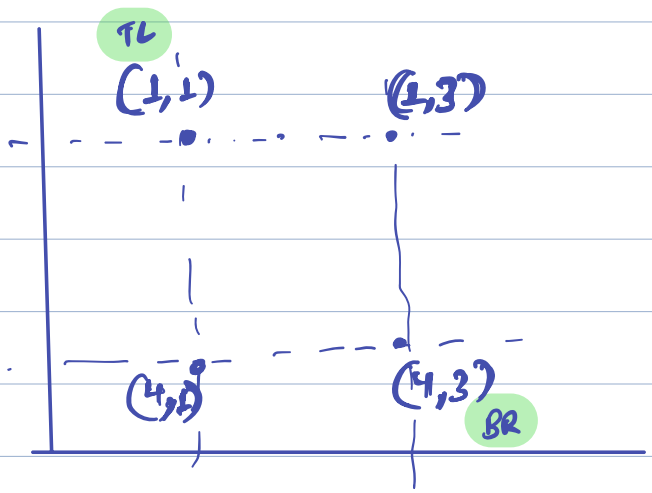
Representing submatrix:

→ mat[6][5]



TL (1,1) TR (1,3)

BL (4,1) BR (4,3)



{Leetcode 304}

Q) Range Sum Query - 2D

↳ Given a mat[N][m] and Q queries, for each query find submatrix sum.

Ex:

	0	1	2	3	4
0	7	1	-6	3	12
1	10	5	1	0	9
2	6	4	-3	8	11
3	13	-8	-5	12	4
4	3	2	1	9	8
5	4	3	-2	6	3

(TL)		(BR)		
x1	y1	x2	y2	
2	1	4	3	: 20
3	2	5	4	: 35

Idea:

↳ for every query iterate on the submatrix and get the sum.

T.C: O(N*m*Q) S.C: O(1)

Optimal approach

Psum[i] = Sum(0,i) | Sum(i,j) = Psum[j] - Psum[i-1]

↓

Psum[i][j] = sum{(0,0) - (i,j)}

(0,0) ↑ TL, (i,j) ↑ BR

arr

	0	1	2	3	4
0	7	1	-6	3	12
1	10	5	1	0	9
2	6	4	-3	8	11
3	13	-8	-5	12	4
4	3	2	1	9	8
5	4	3	-2	6	3

psum

	0	1	2	3	4
0	7			5	
1			18		
2					
3			35		
4					
5		50			

//2D Prefix Sum

arr

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

① Apply row-wise Prefix Sum →

	0	1	2
0	a	a+b	a+b+c
1	d	d+e	d+e+f
2	g	g+h	g+h+i

Apply Col-wise Prefix Sum ↓

Psum

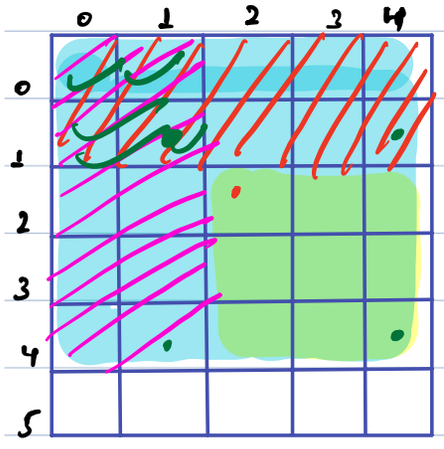
	0	1	2
0	a	a+b	a+b+c
1	a+d	a+b+d+e	a+b+c+d+e+f
2	a+d+g	a+b+d+e+g+h	a+b+c+d+e+f+g+h+i

=

	0	1	2
0	a	a+b	a+b+c
1	a+d	a+b+d+e	a+b+c+d+e+f
2	a+d+g	a+b+d+e+g+h	a+b+c+d+e+f+g+h+i

T.C: O(N*M)

//Solving the query



query

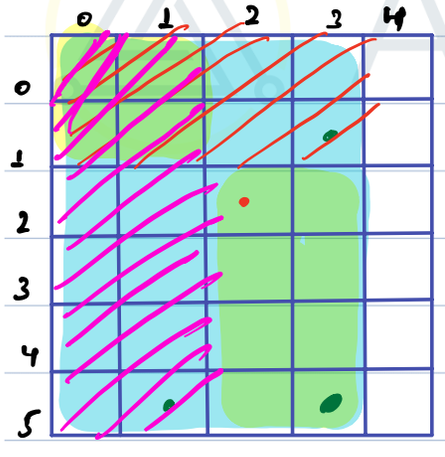
TL (2,2)

BR (4,4)

Psum[4][4] - Psum[1][4] - Psum[4][1]

+ Sum[(0,0) - (1,1)]

+ Psum[1][1]



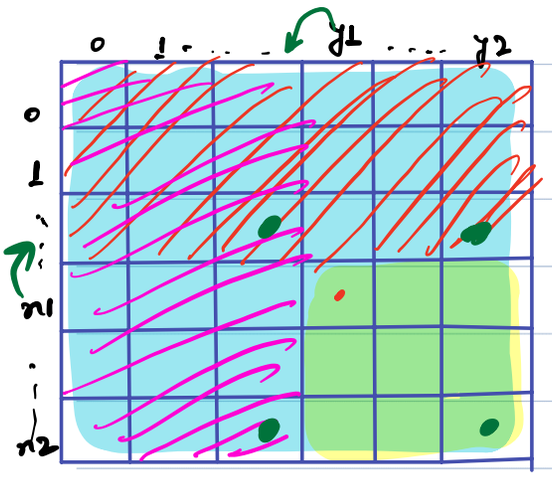
query

TL (2,2)

BR (5,3)

Psum[5][3] - Psum[1][3] - Psum[5][1]

+ Psum[1][1]



Query

(x_1, y_1) TL

(x_2, y_2) BR

$Psum[x_2][y_2] - Psum[x_1-1][y_2]$

$Psum[x_2][y_1-1] + Psum[x_1-1][y_1-1]$

Psum

i \ j	0	1	2
0	a	a+b	a+b+c
1	d	d+e	d+e+f
2	g	g+h	g+h+i

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

a	a+b	a+b+c

//Pseudo Code

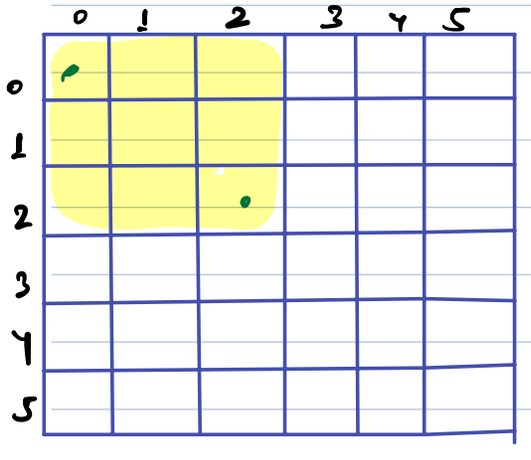
```
Class NumMatrix {
    int[][] Psum;                                   // creating the pref in sum
    Public NumMatrix (int[][] matrix) {
        Psum = new int[matrix.length][matrix[0].length];
        int n = matrix.length;
        int m = matrix[0].length;

        for(int i=0; i<n; i++) {
            for(int j=0; j<m; j++){
                if (j==0){ Psum[i][j] = matrix[i][j]
                else {
                Psum[i][j] = Psum[i][j-1] + mat[i][j];
                }
            }
        }

        for(int j=0; j<m; j++){
            for(int i=1; i<n; i++){
                Psum[i][j] = Psum[i-1][j] + Psum[i][j]
            }
        }
    }
}
```

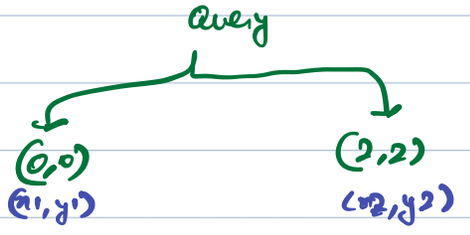
T.C: $O(N*M) + O(Q)$

$\Downarrow$

$O(N*M+Q)$

```
Public int sumRegion (int x1, int y1, int x2,
                      int y2 ){
    int sum = 0;
    sum = sum + Psum[x2][y2];
    if (x1-1 >= 0){
        sum = sum - Psum[x1-1][y2];
    }
    if (y1-1 >= 0){
        sum = sum - Psum[x2][y1-1];
    }
    if (x1-1 >= 0 && y1-1 >= 0){
        sum = sum + Psum[x1-1][y1-1];
    }
    return ans;
}
```





int ans = Psum[x2][y2] - Psum[x1-1][y2]
- Psum[x2][y1-1] + Psum[x1-1][y1-1]

Psum[2][2] - Psum[-1][2] - Psum[2][-1]
+ Psum[-1][-1]

```
for(int j=0; j<m; j++){
    for(int i=1; i<n; i++){
        psum[i][j] = psum[i-1][j] + psum[i][j-1]
    }
}
```

psum

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

	0	1	2
0	a	a+b	a+b+c
1	a+d	a+b+d+e	a+b+c+d+e+f
2	a+d+g	a+b+d+e+g+h	a+b+c+d+e+f+g+h+i

Q) Given mat[n][m], find sum of all submatrix sum.

Ex: mat[6][5] → TL BR

	0	1	2	3	4
0	TL	TL	TL		
1	TL	TL	TL		
2	TL	TL	TL BR	BR	BR
3			BR	BR	BR
4			BR	BR	BR
5			BR	BR	BR

6*5

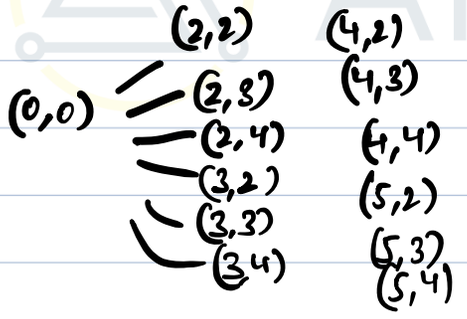
Idea → Contribution technique

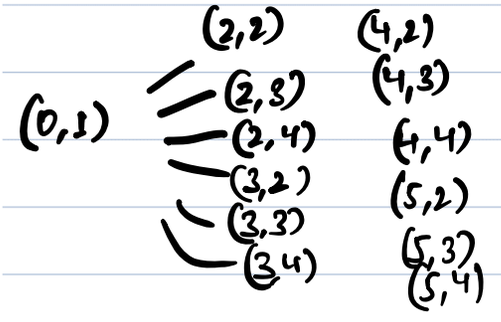
Σ (occ * mat[i][j])

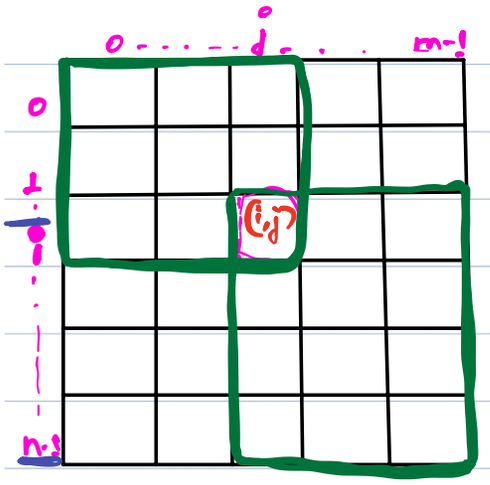
valid TL = 9

valid BR = 12

No. of submatrix in which (2,2) is present = 9x12 = 108







valid TL = (i+1) * (j+1)

valid BR = (n-i) * (m-j)

Total occ of (i,j) = (i+1) * (j+1) * (n-i) * (m-j)

// Pseudo code

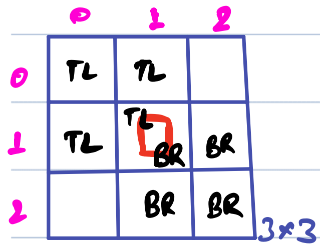
```
int SubmatrixSum(int arr[N][M]){
    int ans = 0;
    for(int i=0; i<N; i++){
        for(int j=0; j<M; j++){
            int occ = (i+1)*(j+1)*(n-i)*(m-j);
            ans = ans + (occ * arr[i][j]);
        }
    }
    return ans;
}
```

T.C: O(N*M)

S.C: O(1)

$2*2 * 2*2 = 16$

$(i+1)*(j+1)*(n-i)*(m-j);$



$(1,1)$ occ $= 4*4 = 16$

Contribution technique on 2D Arrays