1Q) Given N Array elements & Q queries, for each query find sum of elements in given range.

	0	1	2	3	4	5	6	7	8	9
ar[10] :	3	-2	1	4	2	6	2	-4	2	8

Q : 3

$0 <= l <= r < N$

BF:
For every iterate & get sum

TC : $Q * (N)$ SC : $O(1)$

l	r	
4 - 8	:	
3 - 9	:	pf[9] - pf[2]
5 - 7	:	pf[7] - pf[4]

// pf[i] = sum of all elements from 0 - i

	0	1	2	3	4	5	6	7	8	9
ar[10] :	3	-2	1	4	2	6	2	-4	2	8
pf[] :	3	1	2	6	8	14	16	12	14	22

// sum of [0 - 8]

4 - 8 : pf[8] = [0 - 3] + sum [4 - 8]

= pf[3] + sum [4 - 8]

$\boxed{\text{sum }[4-8] = pf[8] - pf[3]}$

// sum of all [0 r]

// $\underline{l - r}$: $pf[r] = \underbrace{[0, l-1]}_{pf[l-1]} + [l\ r]$

$pf[l-1] + sum[l\ r]$

$$\boxed{sum[l\ r] = pf[r] - pf[l-1]}$$

Edge Case: l == 0, it's an issue

Q: $[0\ 3]$: $pf[3] - pf[0-1]$

: $pf[3] - pf[-1]$

$[0\ 3]$: $pf[3]$

$\underline{l - r}$:

```
if( l == 0):
    pf[r]
else:
    pf[r] - pf[l-1]
```

TC: $O(N + Q * 1)$

// How to construct pf[]

// sum of [0 i]

$pf[i] = sum[0, i-1] + ar[i]$

$pf[0] = ar[0]$

```
i = 1; i < N; i++){
    pf[i] = pf[i-1] + ar[i]
}
```

Edge i == 0, corr

SC: $O(N)$ → If we can modify arr[], use it to constru pf[]

→ SC: $O(1)$

2Q) Given N Array elements & Q queries, for each query check if given subarray is increasing order or not?

	0	1	2	3	4	5
Eg1: ar[5] :	1	7	3	4	9	9

constraints

$0 <= l < r < N$

{ To avoid single element Cases }

q_1 : [2 4] : True

q_2 : [1 3] : False

q_3 : [2 5] : False

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[13] :	1	3	4	6	2	8	6	5	10	12	14	16	9

q : l r

q_1 : [0 3] : True

q_2 : [4 7] : False

q_4 : [7 11] : True

q : [l r] : → [l, r-1]

BF: fro every query iterate & check If increasing → TC : O(N)

[7 → 11] → [7 10] iterate & check

7 → 8 ✓

8 → 9 ✓

9 → 10 ✓

10 → 11 ✓

11 → 12

out of bounds

In general → [l, r-1] if increasing

```
i = l; i < r; i++){
    if( ar[i] >= ar[i+1]){
        return False;
    }
}
return True
```

Eg2:

	0	1	2	3	4	5	6	7	8	9	10	11	12
ar[12] :	1	3	4	6	2	8	6	5	10	12	14	16	9
	1	1	1	0	1	0	0	1	1	1	1	0	1/0

	check.	expected sum	Actual Sum	
q_1 : [0 3] →	[0 2] →	3	3	
q_2 : [4 7] →	[4 6]	3	1	✗
q_3 : [0 4] →	[0 3]	4	3	✗
q_4 : [7 11] →	[7 10]	4	4	✓
q_5 : [l r] →	[l r-1]	r - l	[l, r-1]	

pf[r-1] - pf[l-1]

Pseudo Code

1) Based on inc/dec conv arr[] → 1/0

2) Apply pf[] ✓ ⇒ expected   Actual fr

3) Now for Query L–R = r–l == pf[r-1] - pf[l-1]

4) Edge Case : If l==0

TC : O(N + N + Q * (1))   SC : ↱ O(N) : Extra array
↳ O(1) : Modify array

Q) Given a matrix of size N*M, for each query q find sum of given submatrix ? → part of matrix

↳ for any opposites of a diagonal it is fixed

TL & BR

TL & BR diagonal

Q: a_1, b_1 $\quad a_2, b_2$

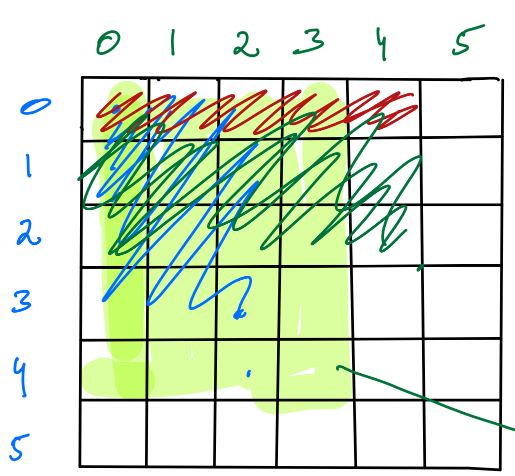
		0	1	2 (b_1)	3 (b_2)	4	5
	0	7	1	-6	3	12	-2
a_1	1	10	5	-2	0	9	4
	2	6	4	-3	8	11	3
	3	13	-8	-5	12	4	6
a_2	4	3	2	1	9	3	9
	5	4	3	-2	6	8	8

Ex: Top left & Bottom right
(1,2) (4,3) : 20

Idea: iterate on entire submatrix & get sum.

```
i = a1 ; i <= a2 ; i++) {
    j = b1 ; j <= b2 ; j++) {
        s = s + mat[i][j]
    }
}
```

TC: Q * [N * M] SC: O(1)



pf[i][j] = sum of all [0,0] - [i,j]

Pf[3][2] = [0,0] - [3,2]

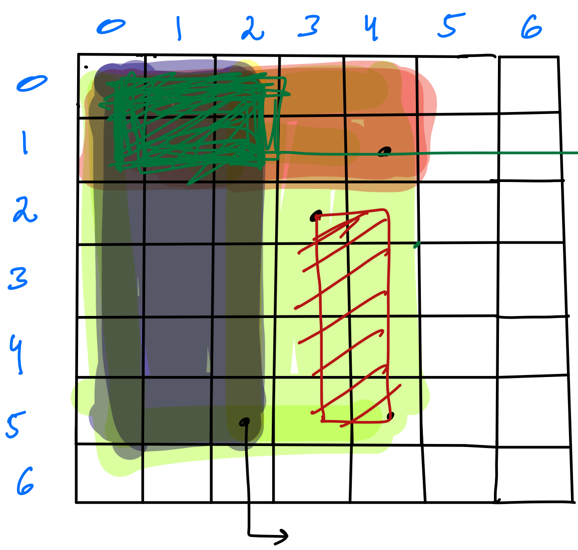
Pf[0][4] = [0,0] - [0,4]

Pf[2,4] = [0,0] - [2,4]

Pf[4,3] = [0,0] → [4,3]

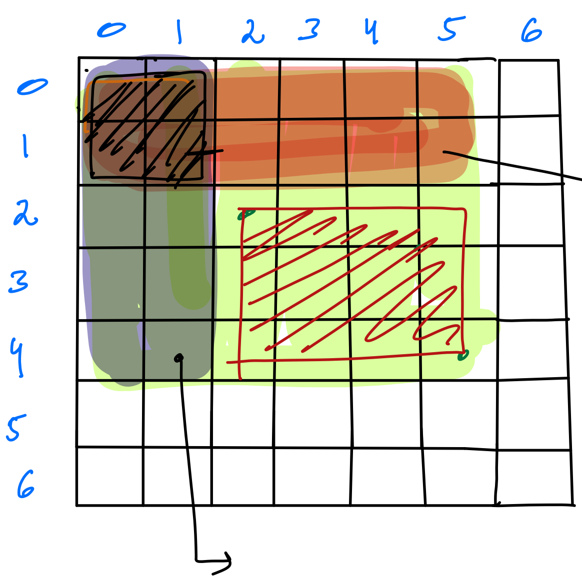
TL BR

11PM



$Q_1: (2,3) \to (5,4)$

$pf[5,4] - pf[5,2] - pf[1,4] + pf[1,2]$



$Q_2: (2,2) \to (4,5)$

$pf[4,5] - pf[4,1] - pf[1,5] + pf[1,1]$

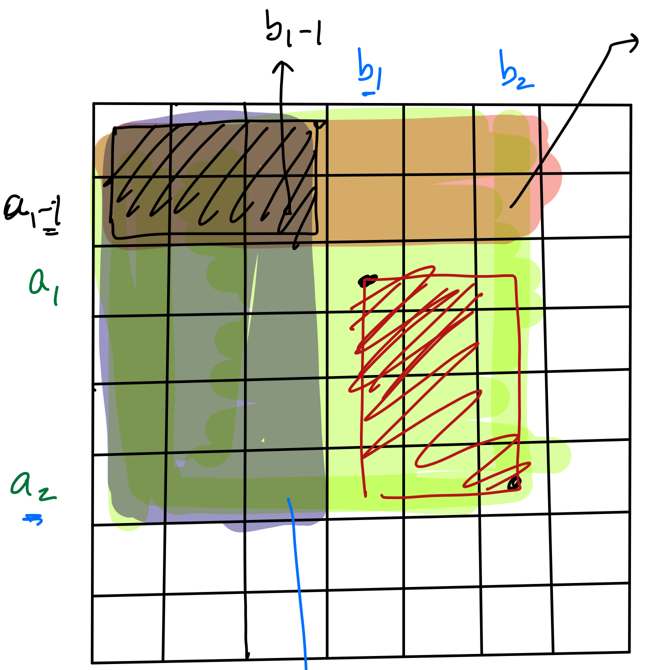
Note: pf[][]

1) Apply pf on every row
2) Apply pf on every col

TC:

$O(N*M + M*N + Q*O(1))$

SC: $O(1)$



a_1-1, b_2

$[a_2, b_1-1]$

Q3: $(a_1, b_1) \to (a_2, b_2)$

$pf[a_2, b_2] - pf[a_2, b_1-1]$

$- pf[a_1-1, b_2] + pf[a_1-1, b_1-1]$

```
// (a1, b1) -> (a2, b2)

sum = pf[a2 b2]
if (b1>0)
{ sum = sum - pf[a2, b1-1]
}
if (a1>0)
{ sum = sum - pf[a1-1, b2]
}
if (a1>0 && b1>0)
{ sum = sum + pf[a1-1, b1-1]
}
```

a_0	b_0	c_0
a_1	b_1	c_1
a_2	b_2	c_2

Along every row take pf[]

a_0	a_0+b_0	$a_0+b_0+c_0$
a_1	a_1+b_1	$a_1+b_1+c_1$
a_2	a_2+b_2	$a_2+b_2+c_2$

Along every column take pf[ ]

	0	1	2
0	a_0	a_0+b_0	$a_0+b_0+c_0$
1	a_0+ a_1	a_0+b_0 a_1+b_1	$a_0+b_0+c_0$ $a_1+b_1+c_1$
2	a_0 a_1 a_2	a_0+b_0 a_1+b_1 a_2+b_2	$a_0+b_0+c_0$ $a_1+b_1+c_1$ $a_2+b_2+c_2$

$pf[1][2]$

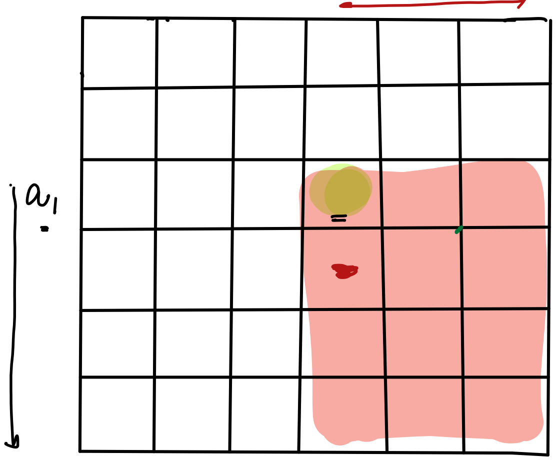
$pf[2][1]$

48) Given a matrix of size N*M, find

Max Submatrix Sum

Ex:

$b_1 : b_2 : [b_1, M-1]$



a_1

$a_2 : [a_1, N-1]$

Idea: Get max of pf[][] ?

wrong?: Because pf[][] will only contain all matrix sums, starting at index [0,0] *

1) Generate all submatrix sum ? get max

TL & BR

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

TL: Any cell in matrix can be TL.

```
a1 = 0; a1 < N; a1++) {
    b1 = 0; b1 < M; b1++) {
        TL: (a1, b1)
        a2 = a1; a2 < N; a2++) {
            b2 = b1; b2 < M; b2++) {
                BR = (a2, b2)
                using pf[][]
                sum, out of all
                sums get max
            }
        }
    }
}
```

// Q) find max submatrix sum of all those matrices which start at row : 2 (index) and end row = 4 (index)

2D kadane's

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

→ find max sum

8 21 -10 5 -22

5	3	2	-1	-10
-1	8	-9	4	-6
5	0	3	2	-6

pf[4][0] − pf[1][0]

row strat = 2 ✓

[8, 21, -10, 5, -22]

→ to get every value we can get pf[] on every column.

row end = 5 ✓

// Compute entire pf col[] mat at start

TC: $O(N^2 * (M + M))$

SC: $O(N * M) + M$

→ We need to construct pf[] for every column →

// rows = 0; rows < N, rows++, ✓

{

// rowe = rows; rowe < N; rowe++; ✓

{

{// We need to M value using pf[]

// On that M value apply kadane's

loop → M value →

TC: O(M)

}

}

	0	1	2	3	4
0	-3	8	-2	-1	10
1	2	5	-12	5	7
2	5	3	2	-1	-10

→ we need to apply column wise to get this sum.

Apply Kadanes:

row: 0 end: 0 : -3 8 -2 -1 10

row: 0 end: 1 : -1, 13, -14, 4, 17

row: 0 end: 2 : 4, 16, -12, 3, 7

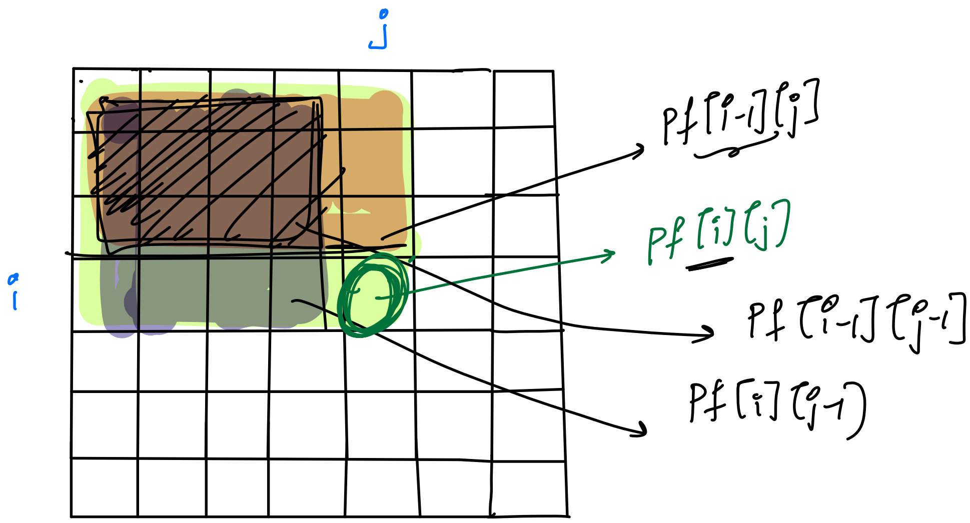
row: 1 end: 1 : 2 5 -12 5 7

row: 1 end: 2 : 7 2 -10 4 -3

row: 2 end: 2 : 5 3 2 -1 -10

// another way to generate pf[]

pf[i][j] = sum of all Element (0,0) → i,j




```
i = 0; i < N; i++) {
    j = 0; j < M; j++) {
        pf[i][j] = ar[i][j]
        if(i > 0) {
            pf[i][j] += pf[i-1][j]
        }
        if(j > 0){
            pf[i][j] += pf[i][j-1]
        }
        if( i > 0 && j > 0) { pf[i][j] -= pf[i-1][j-1] }
    }
}
```

```
j = 0; j < M; j++) {
    i = 0; i < N; i++) {
```