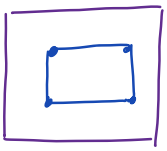
Todays Content:

- Intro to Submatrices
- Submatrix Sum Queries
- Sum of all Submatrices
- Max Submatrix Sum

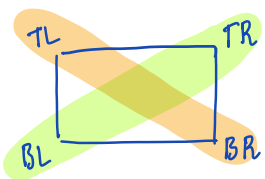
Submatrix: Part of matrix is a Submatrix



Note: a) Single ele is also submatrix

b) Complete matrix is also submatrix

Identify: Any submatrix contains 4 Corners.



We just need Opp corners to get submatrix?

a) Given TL & BR can give us Submatrix

b) Given TR & BL can give us Submatrix

Example:

int mat[6][7] :

	0	1	2	3	4	5	6	7
0	2	4	3	-1	6	5	4	3
1	3	2	6	8	1	10	3	2
2	5	4	3	2	9	3	2	8
3	6	7	3	2	10	11	14	16
4	8	2	3	1	-4	-9	4	2
5	7	6	3	4	7	2	9	11

TL	BR		TR	BL
(1, 2)	(2, 5)	⇒	(1 5)	(2 2)
(2, 4)	(4, 6)	⇒	(2 6)	(4 4)
(3 4)	(5, 6)	⇐	(3 6)	(5 4)
(2 1)	(4 4)	⇐	(2 4)	(4 1)

Con: Given [TL & BR] or [TR & BL] Submatrix can be identified.

In General we use TL & BR to identify a Submatrix.

10) Find sum of all elements in given Submatrix.

Constraints:

$1 \le N * M \le 10^6$

$1 \le mat[i][j] \le 10^6$

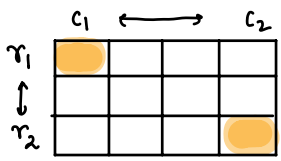
Example: → TL BR sum

(1 2) (3 5) = 68

$(r_1\ c_1)\ (r_2\ c_2)$

	0	1	2	3	4	5	6	7
0	2	4	3	-1	6	5	4	3
1	3	2	6	8	1	10	3	2
2	5	4	3	2	9	3	2	8
3	6	7	3	2	10	11	14	16
4	8	2	3	1	-4	-9	4	2
5	7	6	3	4	7	2	9	11

Idea: Iterate on submatrix, calculate & print sum



Pseudo Code:

```
long SubSum(int mat[][], int r1, int c1, int r2, int c2){
    long sum = 0;                          TC: O(N*M)
    for(int i = r1; i <= r2; i++){         SC: O(1)
        for(int j = c1; j <= c2; j++){
            sum = sum + mat[i][j]
        }
    }
    return sum;
}
```

2Q) Given mat[N][M] and Q queries.

For each query: Given TL & BR Calculate & Print sum of all elements in given submatrix

Note: TL is Top left BR is Bottom right of Submatrix

Constraints:

$1 <= N*M <= 10^6$

$1 <= Q <= 10^5$

$1 <= mat[i][j] <= 10^6$

Ex: mat[4][5]

	0	1	2	3	4
0	3	2	4	1	6
1	-1	4	3	2	4
2	2	7	6	3	2
3	1	2	7	8	1

Queries: Q = 3

TL		BR		
r_1[]	c_1[]	r_2[]	c_2[]	sum
1	1	3	3	42
1	2	2	4	20
2	1	3	1	9

Q mat[3][4]

	0	1	2	3
	r_1	c_1	r_2	c_2
0 →	1	1	3	3
1 →	1	2	2	4
2 →	2	1	3	1

Ideal: For every given query iterate on submatrix calculate sum & print.

TC: $Q * \{N*M\} \Rightarrow Q*N*M$

SC: O(1)

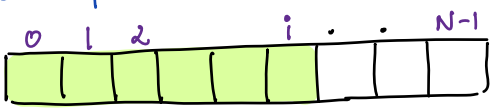
```
void Sum(int mat[N][M], int Q, int r1[], int c1[], int r2[], int c2[]){

    Code TODO:

}
```

Hint: Similar question is done in 1D array & we optimized using

pSum[i] = Sum of all subarray from [0 to i]



TL: Top left
BR: Bottom Right

Extend 2D matrices

pmat[i][j] = Sum of all submatrix ele from [0 0] (TL) to [i j] (BR)

Ex:

mat[3][4] ⟶ Pmat[3][4]

	0	1	2	3
0	3	2	4	1
1	-1	4	3	2
2	2	7	6	3

	0	1	2	3
0	3	5	9	10
1	2	8	15	18
2	4	17	30	36

Pmat[1 0] = sum of all elem in submatrix [0, 0] → [1 0]

Construct:

	0	1	2	3
0	3	2	4	1
1	-1	4	3	2
2	2	7	6	3

Apply cumm sum every row

	0	1	2	3
0	3	5	9	10
1	-1	3	6	8
2	2	9	15	18

Apply cum sum every col

	0	1	2	3
0	3	5	9	10
1	2	8	15	18
2	4	17	30	36

	0	1	2	3
0	3	2	4	1
1	-1	4	3	2
2	2	7	6	3

Apply cumm sum every col

	0	1	2	3
0				
1				
2				

Apply cumm sum every row

	0	1	2	3
0	3	5	9	10
1	2	8	15	18
2	4	17	30	36

Code: Given mat[N][M]

1. long pmat[N][M] ⟶ TODO: Stay back

1. Copy mat[][] ⇒ pmat[][] ⟶ TC: O(N*M)
2. Apply pSum on every row in pMat[][] ⟶ TC: O(N*M)
3. Apply pSum on every col in pMat[][] ⟶ TC: O(N*M)

TC: O(N*M) SC: O(N*M)

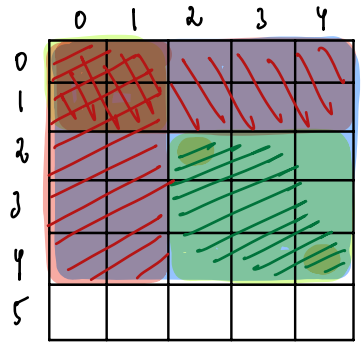
Queries Using Pmat[][]

mat[6][5]

Assume Pmat[6][5] Created

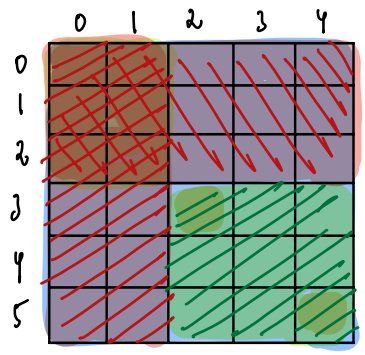
Queries

TL (2, 2) BR (4, 4)



pmat[i][j] = Sum of all elements from [0, 0] (TL) to [i, j] (BR)

Sum = Pmat[4, 4] - Pmat[4, 1] - Pmat[1, 4] + Pmat[1, 1]



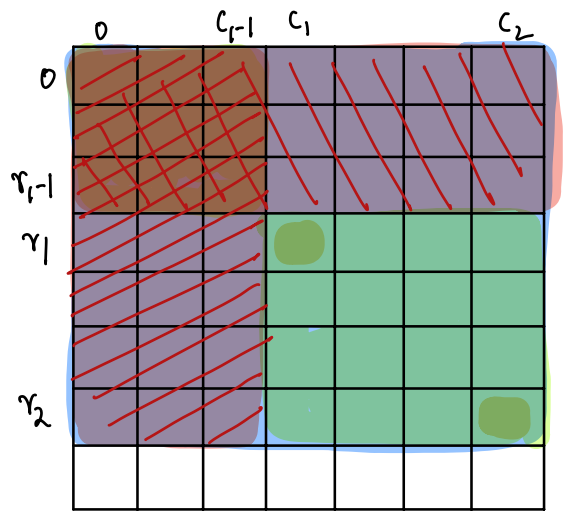
TL (3 2) BR (5, 4)

pmat[i][j] = Sum of all elements from [0, 0] (TL) to [i, j] (BR)

Sum = Pmat[5, 4] - Pmat[5, 1] - Pmat[2 4] + Pmat[2 1]

// Generalized Expression

: mat[N][M] assume Pmat[N][M] is already Constructed.



TL (r_1 c_1) BR (r_2 c_2)

```
long sum = Psum[r2 c2]
if( c1 != 0){
    sum = sum - Psum[r2 c1-1]   // c1=0 Error
}
if(r1 != 0){
    sum = sum - Psum[r1-1, c2]   // r1=0 Error
}
if( r1 != 0 && c1 != 0){
    sum = sum + Psum[r1-1, c1-1]   // r1=0, c1=0
                                   Error
}
```

Eg:

	0	1	2	3	4
0	3	2	4	1	6
1	-1	4	3	2	4
2	2	7	6	3	2
3	1	2	7	8	1

Pfmat[][]

	0	1	2	3	4
0	3	5	9	10	16
1	2	8	15	18	28
2	4	17	30	36	48
3	5	20	40	54	67

Queries:

if $c_1 != 0$ if $r_1 != 0$ if $r_1 != 0$ && $c1 != 0$

$r_1\ c_1\ \ r_2\ c_2 \rightarrow$ Pmat[r_2 c_2] - Pmat[r_2 c_1-1] - Pmat[r_1-1, c_2] + Pmat[r_1-1, c_1-1]

Pmat[3 3] - Pmat[3, 0] - Pmat[0, 3] + Pmat[0 0]

(1, 1) (3, 3) = 54 - 5 - 10 + 3 = 42

if $c_1 != 0$ if $r_1 != 0$ if $r_1 != 0$ && $c1 != 0$

$r_1\ c_1\ \ r_2\ c_2$ Pmat[3 4] - Pmat[3 -1] - Pmat[1 4] + Pmat[1 -1]

(2, 0) (3 4) = 67 - 28 = 39

```
void Sum(int mat[][], int Qmat[][]) {
    int N = mat.length;
    int M = mat[0].length
    long pmat[N][M];
    TODO: Construct pmat[][]                          ---> O(N*M)
    int Q = Qmat.length;                                          } TC: O(N*M + Q)
    for(int i=0; i<Q; i++) {                          ---> O(Q)   } SC: O(N*M)
        // ith Query:
        int r1 = Qmat[i][0]
        int c1 = Qmat[i][1]
        int r2 = Qmat[i][2]
        int c2 = Qmat[i][3]
        long sum = Psum[r2 c2]
        if( c1 != 0){
            sum = sum - Psum[r2 c1-1]
        }
        if(r1 != 0){
            sum = Sum - Psum[r1-1, c2]
        }
        if( r1 != 0 && c1 != 0){
            sum = sum + Psum[r1-1, c1-1]
        }
        print(sum);
    }
}
```

Qmat:

0	1	2	3
r_1	c_1	r_2	c_2

i^{th} Query:

0	1	2	3
r_1	c_1	r_2	c_2

2Q) Given mat[N][M] find sum of all submatrices sum.

Eg: $\begin{bmatrix} 3 & 1 \\ -1 & -2 \\ 2 & 4 \end{bmatrix}$ = $[3]^3$ $[3\ 1]^4$ $\begin{bmatrix} 3 \\ -1 \end{bmatrix}^2$ $\begin{bmatrix} 3 & 1 \\ -1 & -2 \end{bmatrix}^1$ $\begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix}^4$ $\begin{bmatrix} 3 & 1 \\ -1 & -2 \\ 2 & 4 \end{bmatrix}^7$

$[1]^1$ $\begin{bmatrix} 1 \\ -2 \end{bmatrix}^{-1}$ $\begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix}^3$ $[-1]^{-1}$ $[-1\ -2]^{-3}$ $\begin{bmatrix} -1 \\ 2 \end{bmatrix}^1$ $\begin{bmatrix} -1 & -2 \\ 2 & 4 \end{bmatrix}^3$

$[2]^2$ $[2\ 4]^6$ $[4]^4$ $[-2]^{-2}$ $\begin{bmatrix} -2 \\ 4 \end{bmatrix}^2$

→ Sum of all submatrices = 36

Idea = Contribution Technique = $3*6 + 1*6 + -1*8 + -2*8 + 2*6 + 4*6$

= $18 + 6 - 8 - 16 + 12 + 24 = 36$

↳ Add Contribution of each element to get final ans.

In how many submatrices cell 1,2 is present ?

Construct

TL = 6 BR = 9 Total Count = 54

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

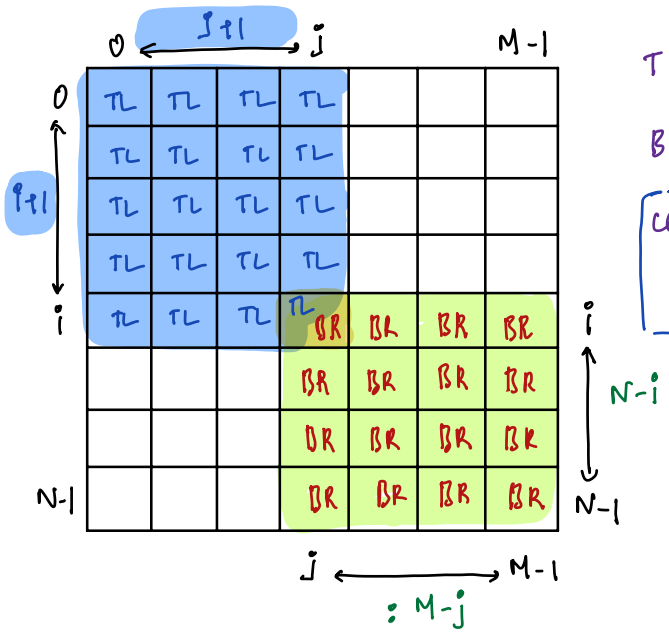
(0 0) (0 1) (0 2) (1 0) (1 1) (1 2) (1 2) (2 4) (1 3) (3 2) (1 4) (3 3) (2 2) (3 4) (2 3)

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

In how many submatrices cell 3,2 is present ?.

TL = 12 BR = 9 Total Count = 108

mat[N][M] In how many submatrices cell (i,j) is present



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

BR = (N-i)(M-j)

cells, (i,j) =

TL * BR = (i+1)(j+1)(N-i)(M-j)

```
long allsum(int mat[][]){ TC: O(N*M)  SC: O(1)
    int N = mat.length, M = mat[0].length
    long sum = 0
    for(int i=0; i<N; i++){
        for(int j=0; j<M; j++){
            long TL = (i+1)(j+1)
            long BR = (N-i)(M-j)
        }   sum = sum + mat[i][j]*TL*BR
    }
    return sum
}
```

38] Given row-wise & col wise sorted matrix of mat[N][M]

Find Manimum submatrix sum

Eg1:

	0	1	2	3
0	-20	-16	-4	8
1	-10	-8	2	14
2	-1	6	21	30
3	5	7	28	42

ans = 142

Eg2:

	0	1	2	3
0	-20	-16	-4	-1
1	-10	-8	-2	5
2	-4	2	4	8

ans = 15

Eg3:

	0	1	2
0	-50	-40	-30
1	-35	-20	-15
2	-19	-14	-3

ans = -3

Obs:

Given mat[N][M]

: For ans submatrix

TL : Take matrix cell as TL

BB : [N-1, M-1]

Eg: mat[2][3]

	0	1	2
0	-7	2	4
1	4	7	10

TL	BR	Sum
0 0	1 2	20
0 1	1 2	23
0 2	1 2	14
1 0	1 2	21
1 1	1 2	17
1 2	1 2	10

ans = 23

Pseudo Code:

```
long maxSub(int mat[][]){ TC : O(N*M)  SC: O(N*M)
    int N = mat.length;
    int M = mat[0].length;
    int ans = INT_MIN
    long pmat[N][M] // TODO construct ----> O(N*M)

    for(int i=0; i<N; i++){                  } ----> O(N*M)
        for(int j=0; j<M; j++){
            // TL {r1 = i, c1 = j}  // BR - {r2 = N-1, c2 = M-1}
            // Calulate Submatrix Sum:
               int r1 = i
               int c1 = j
               int r2 = N-1
               int c2 = M-1
               long sum = psum[r2 c2]
               if( c1 != 0){
                  sum = sum - psum[r2 c1-1]
               }
               if(r1 != 0){
                  sum = sum - psum[r1-1, c2]
               }
               if( r1 != 0 && c1 != 0){
                  sum = sum + psum[r1-1, c1-1]
               }
               if( sum > ans) { ans = sum}
        }
    }
} return ans;
```

Given mat[N][M] Construct pMat[N][M]

Eg: mat[3][4]

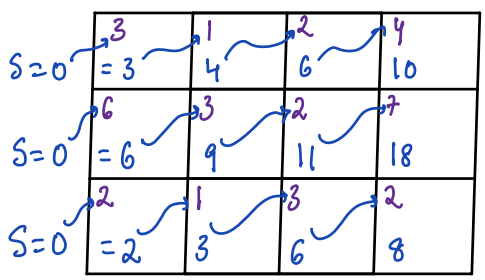
	0	1	2	3
0	3	1	2	4
1	6	3	2	7
2	2	1	3	2

Step1: Copy mat[3][4] → pmat[3][4]

	0	1	2	3
0	3	1	2	4
1	6	3	2	7
2	2	1	3	2

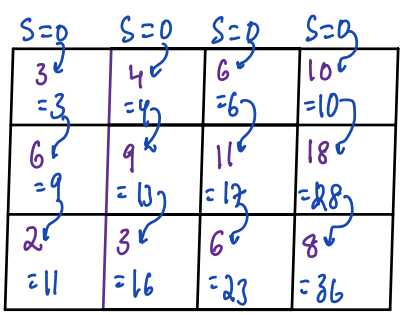
```
for(int i=0; i<N; i++){
    for(int j=0; j<M; j++){
        pmat[i][j] = mat[i][j]
    }
}
```

Step2: Apply Cummlative in every row




```
for(int i=0; i<N; i++){
    long sum=0 // ith row
    for(int j=0; j<M; j++){
        sum = sum + pmat[i][j]
        pmat[i][j] = sum
    }
}
```

Step3: Apply Cummlative in every col

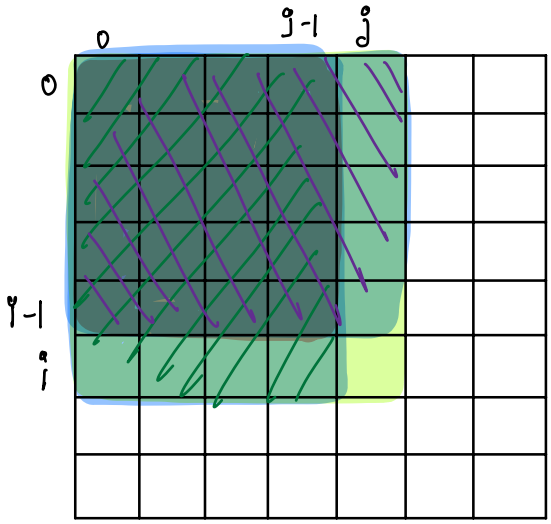



```
for(int j=0; j<M; j++){
    long sum=0 // jth col
    for(int i=0; i<N; i++){
        sum = sum + pmat[i][j]
        pmat[i][j] = sum
    }
}
```

// Construct pMat[][] using formula

pSum[i] = pSum[i-1] + ar[i]

pMat[i,j] = pMat[i,j-1] + pMat[i-1,j] - Pmat[i-1,j-1] + mat[i][j]



// Given mat[N][M]

```
long pmat[N][M];
for(int i=0; i<N; i++){ ⟶ TC: O(N*M)  SC: O(N*M)
    for(int j=0;j<M;j++){
        if(j!=0){ pmat[i][j] += pMat[i,j-1] }
        if(i!=0){ pmat[i][j] += pMat[i-1,j] }
        if(i!=0 && j!=0){ pmat[i][j] += -Pmat[i-1,j-1] }
        pmat[i][j] += mat[i][j]
    }
}
```