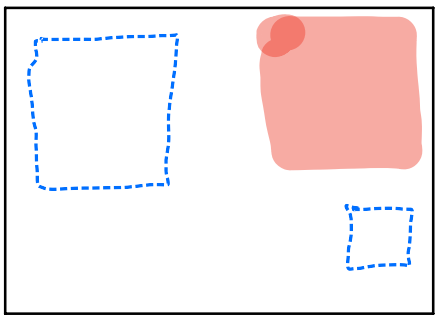
Arrays 2D

Today's Content

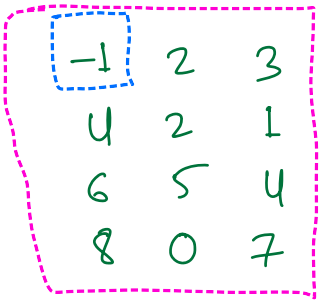
- Introduction to submatrix
- Submatrix sum queries
- Sum of all submatrix sum
- Maximum sum submatrix {Hint} for sorted array

Submatrix : Part of a matrix.
Any rectangle inside the matrix.

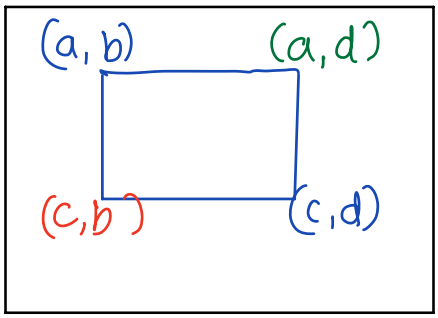


Note : A single element is also a submatrix

Entire matrix is also a submatrix



Represent a submatrix



Points to rep submatrix

- TL (top left)
- BR (bottom right)

int mat[5][6]

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

TL = (2,1)

BR = (4,4)

Q> Given a matrix of size R*C and Q queries, For each query, Find sum of given submatrix

Note: TL = top left & BR = bottom right.

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

Queries

TL	BR	
(2,1)	(4,2)	→ 26
(1,1)	(3,3)	→ 33

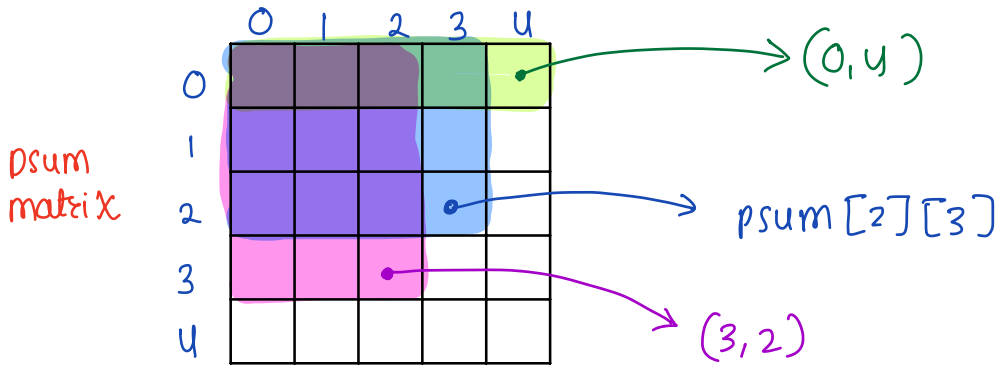
Bruteforce : TL $r_1 c_1$ BR $r_2 c$

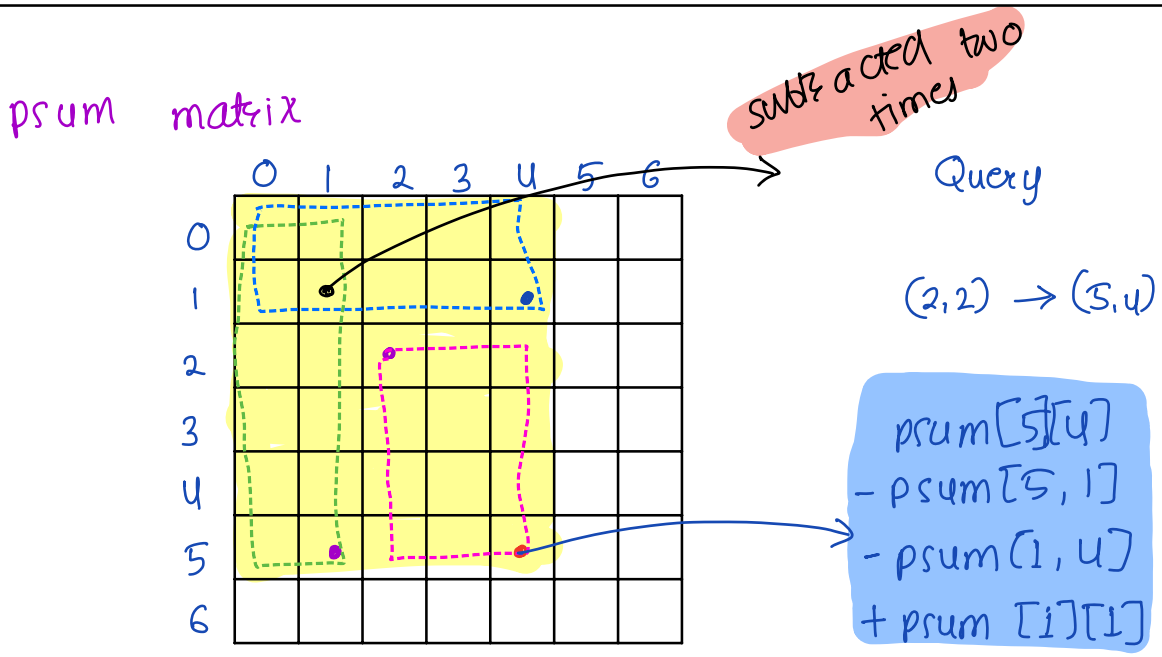
TC: O(QRC)

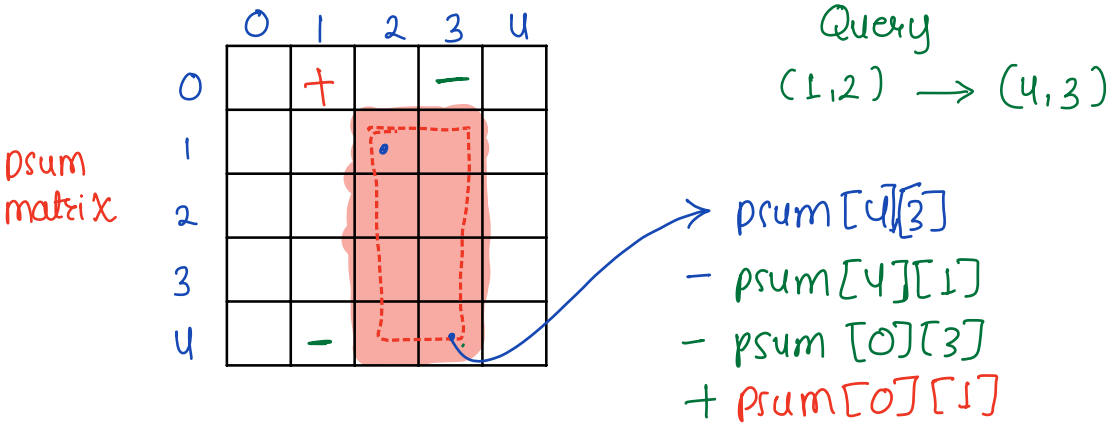
```
for (q=0 ; q< Q ; q++) {
    TL {r1,c1}
    BR {r2 c2}
    total = 0
    for (r=r1 ; r <= r2 ; r++) {
        for (c = c1 ; c<=c2 ; c++) {
            total += A[r][c]
        }
    }
}
```

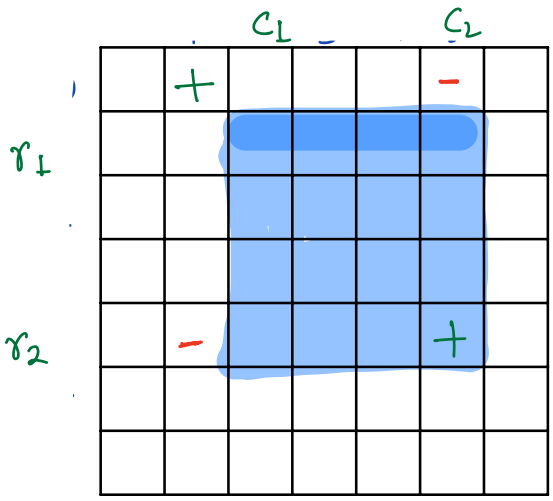
1D psum[i] = sum of all elements from 0 – i^{th}

2D psum[i][j] = sum of all elements from (0,0) to (i,j)


psum matrix
0 1 2 3 4
0
1
2
3
4
(0,4)
psum[2][3]
(3,2)


psum matrix
subtracted two times
Query
(2,2) → (5,4)
0 1 2 3 4 5 6
psum[5][4]
- psum[5,1]
- psum[1,4]
+ psum[1][1]


Query
(1,2) → (4,3)
psum matrix
0 1 2 3 4
+
-
-
psum[4][3]
- psum[4][1]
- psum[0][3]
+ psum[0][1]



psum [r2] [c2]
- psum [r1-1] [c2]
- psum [r2] [c1-1]
+ psum [r1-1] [c1-1]

```
total = psum [r2] [c2]
if (r1>0) {
|     total -= psum [r1-1] [c2]
}
if (c1>0) {
|     total -= psum [r2] [c1-1]
}
if(r1>0 && c1>0) {
|     total += psum [r1-1] [c1-1]
}
```

To build psum matrix

1. prefix sum of each row
2. prefix sum of each col

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

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

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

final psum mat →

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

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

psum row →

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

psum col ↓

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

TC : O(R x C)

TC of optimized solution for submatrix sum queries

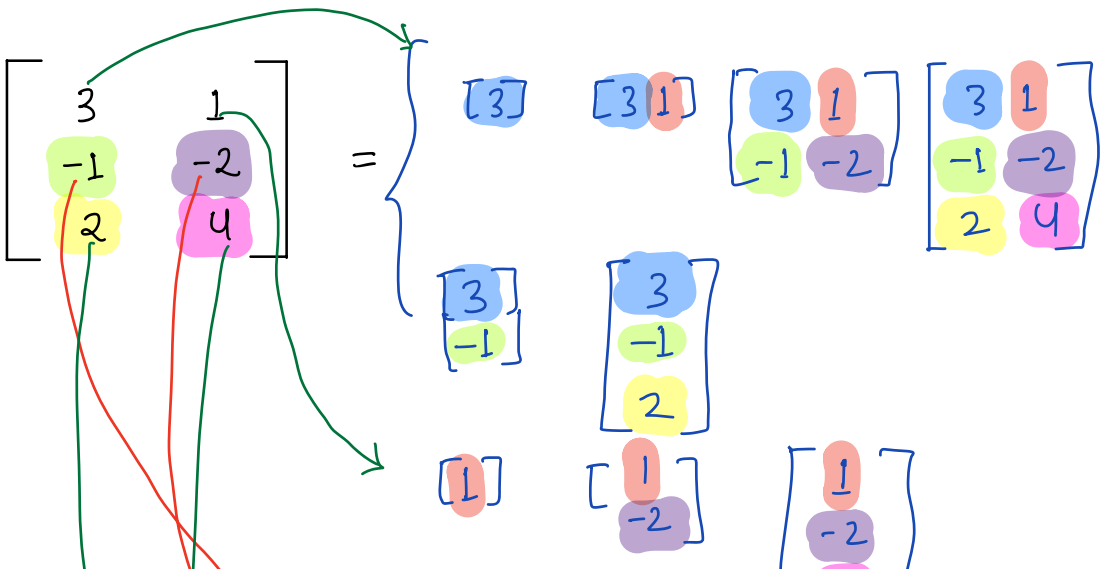
$O(Q+RC)$

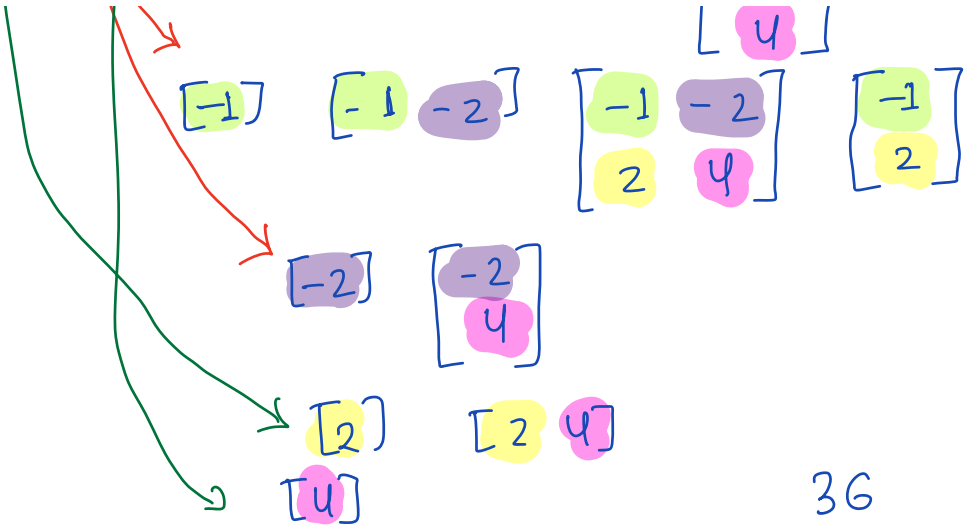
SC : $O(R \times C)$

```
M[R][C]
TLR[Q]
TLC[Q]
BRR[Q]
BRC[Q]

P = [][]   // init
// Build psum mat.

for (r=0 ; r<R ; r++) {
    for(c = 1 ; c<C ; c++) {
        P[r][c] = P[r][c-1] + M[r][c]
}   }

for (c=0 ; c<C ; c++ ) {
    for(r = 1 ; r<R; r++ ) {
        P[r][c]+= P[r-1][c]
}   }

for (q=0; q<Q ; q++) {
    r1 = TLR[q] , r2 = BRR[q]
    c1 = TLC[q] , c2 = BRC[q]
    // plug in the query formula.
```

```
total = p[r2][c2]
if (r1 > 0) {
    total -= p[r1-1][c2]
}
if (c1 > 0) {
    total -= p[r2][c1-1]
}
if (r1 > 0 && c1 > 0) {
    total += p[r1-1][c1-1]
}
}
```

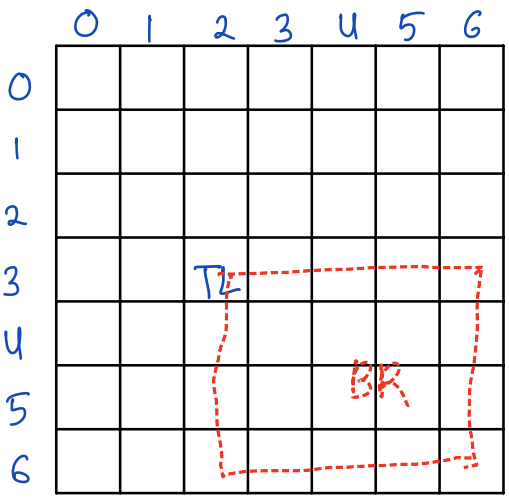
Q> Given a matrix of size R*C. Calculate sum of all submatrix sums.





Brute force.

- Fix a TL point and generate all BR points
- Use sum formula to get sum b/w(TL, BR)



Pseudo

$TC = R^2C^2$

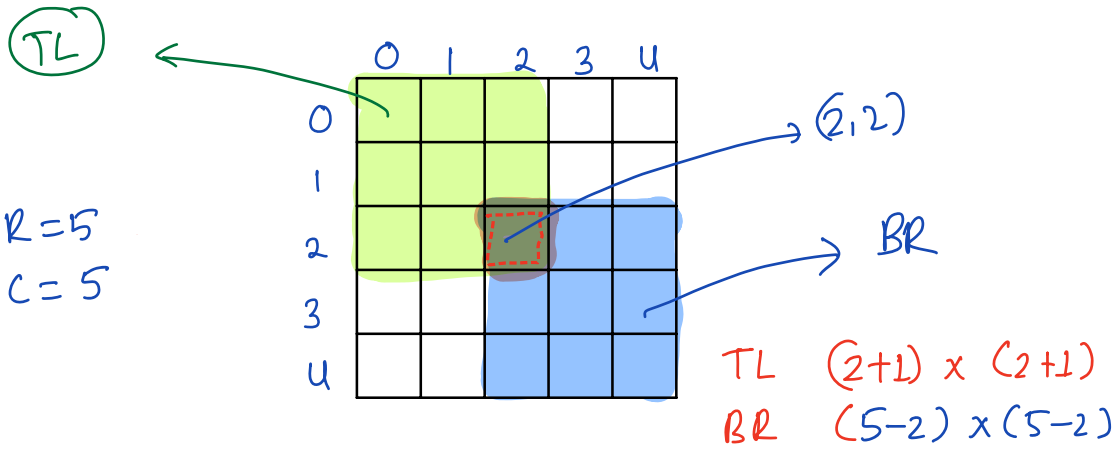
$SC = RC$

```
total = 0
for (r1 = 0 ; r1 < R ; r1++) {
    for (c1 = 0 ; c1 < C ; c1++) {
        for (r2 = r1 ; r2 < R ; r2++) {
            for (c2 = c1 ; c2 < C ; c2++) {
                total += // we psum logic
            }
        }
    }
}
```

Contribution

3 x 6	→	18
1 x 6	→	6
-1 x 8	—	-8
-2 x 8	—	-16
+2 x 6	—	12
4 x 6	→	24
		36

In How many matrices will an element appear

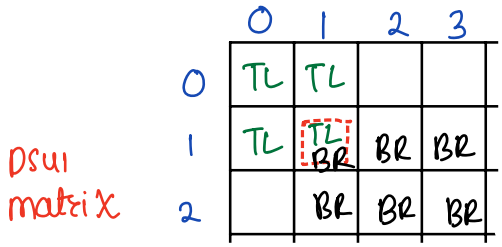


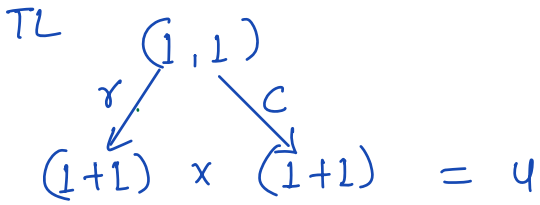
Opts of TL point = X

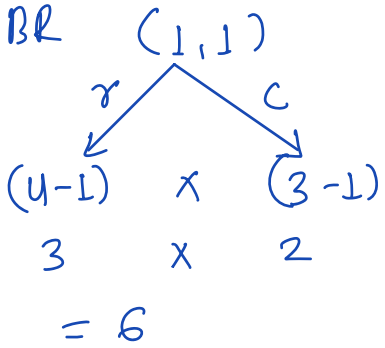
Opts of BR point = Y

Total options = X x Y

Contribution = X x Y x M[i][j]







r, c (R, C)

X Pos for TL = (r+1) x (c+1)

Pos for BR = R-r x C-c

Y

TC : RXC

SC : 1

Q> Given row wise and col wise sorted matrix.
Find max submatrix sum.

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

fix this as BR

Doubt session

	L		R
	1	2	3
	1	3	6

P[R] - P[L-1]

get(P, R) - get(P, L-1)

```
get ( P , idx ) {
  if ( idx >=0 && idx < P.length)
      return P[idx]

  return 0
}
```