Today's Content

1) Sum of all submatrices
2) Max submatrix sum
3) Rain water trapping
4) Min swaps to bring all Elements $x = k$ together
   → Doubts Session
5) 2D Kadane's one again.

18) Given a matrix of size N*M, calculate sum of all Submatrix Sum

Ques1:

$$\begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix}$$

[3] [-1] [-2] [1] [3 -1; -2 1]

[3 -1] [3 -2] [-2 1] [-1 1]

} 4

⇒ 3*4 + -1*4 + -2*4 + 1*4 = 4

12 + -4 - 8 + 4 = 4

Eg: [1 2 3] ⇒ [1] [2] [3] [1 2] [2 3] [1 2 3]

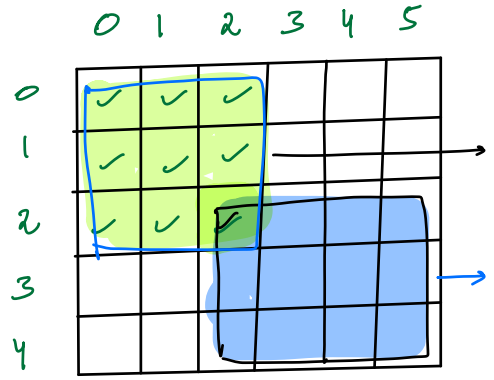
3 4 3

// Sol1: Generate all submatrix & add sum

TC = O(N²M²), we will use premat[] SC: O(1) → O(N*M)

// Idea1: Sum of all subarr sums

= [Number of time mat[i][j] comes in all submat] * mat[i][j]

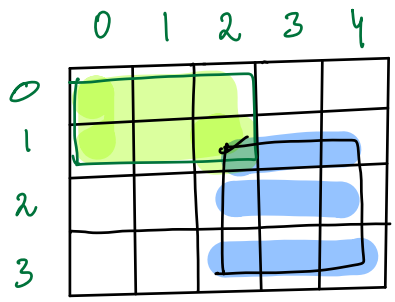


submatrix

TL = 9

BR = 12 = 108

Query: 2 1 2 TL = 6 BR = 9, Total = 54



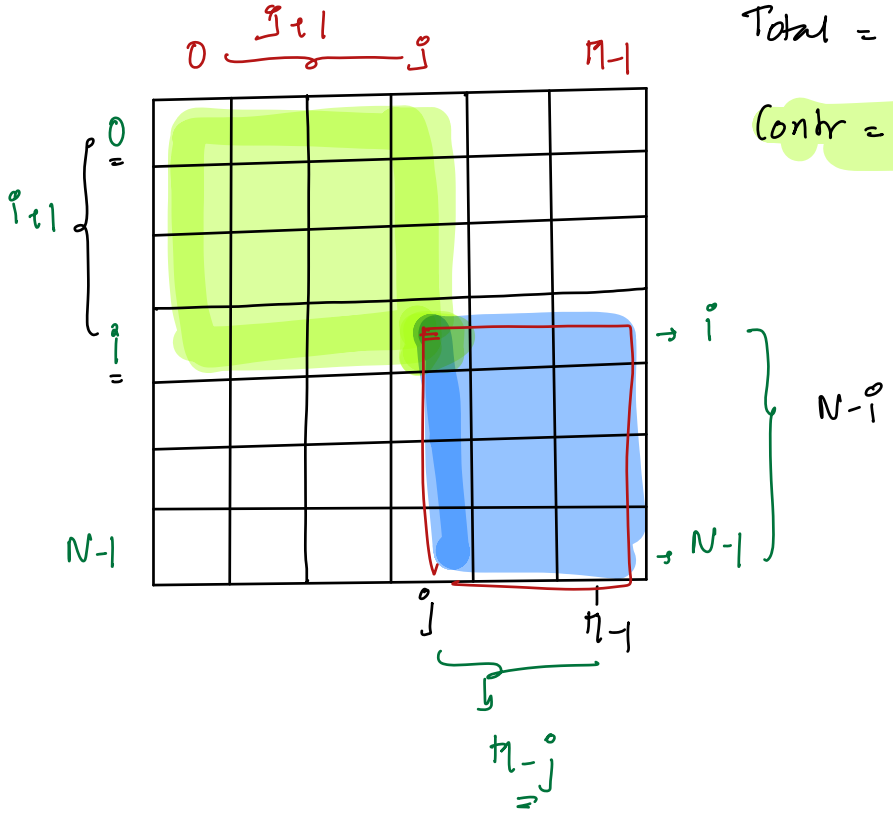
✓ TL = (i+1) * (j+1)

✓ BR = (N-i) * (M-j)

Total = TL * BR

Contr = TL * BR * mat[i][j]

Query: 3 & Query 4



```
ans = 0;
i = 0; i < N; i++) {
    j = 0; j < M; j++) {
        TL = (i+1) * (j+1)
        BR = (N-i) * (M-j)
        Total = TL * BR
        ans += TL * BR * mat[i][j]
    }
}
```

TC: $O(N*M)$ ✓

SC: $O(1)$

→ overflows?

→ Tell you get % m

28) Given a matrix every row is sorted & every column is sorted find max submatrix sum.

Eg:

	0	1	2	3
0	-20	-16	-4	8
1	-10	-8	12	14
2	-1	6	21	30
3	5	7	28	42

→ Solutions

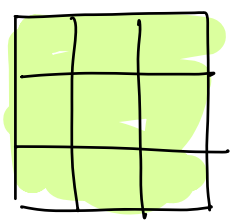
1) Find all submatrix & get max

TC: $O(N^2 M^2)$ SC: $O(NM)$

2) 2D Kadane's → [Doubts Session]

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

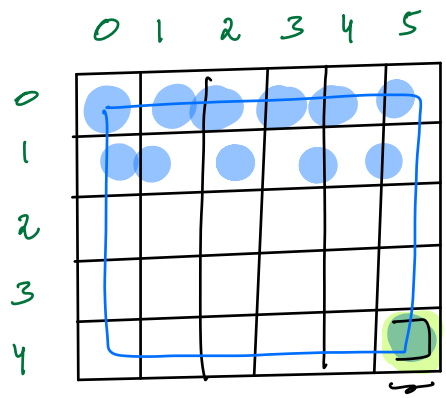
Eg1: All +ve Positive:



Eg2: All -ve Elements : Max element

-50	-40	-30	-20
-41	-30	-23	-18
-35	-25	-19	-16
-29	-20	-18	-14

-14 → max submatrix sum



Ans submatrix should contain

[N-1][M-1]

ans submatrix

TL — BR

(N-1)(M-1)

Take every cell as Top-left & see which cell gave sum.

// Pseudo code :

ans = mat[0][0]

```
for(i=0; i<N; i++){
    for(j=0; j<M; j++){
        TL = (i, j)   BR = (N-1, M-1)
            ↳ get sum using pf[][]
        ans = max(ans, sum)
    }
}
```

TC: O(N*M + N*M) — from TL; O(1) to construct pf[][]

SC: O(N*M)

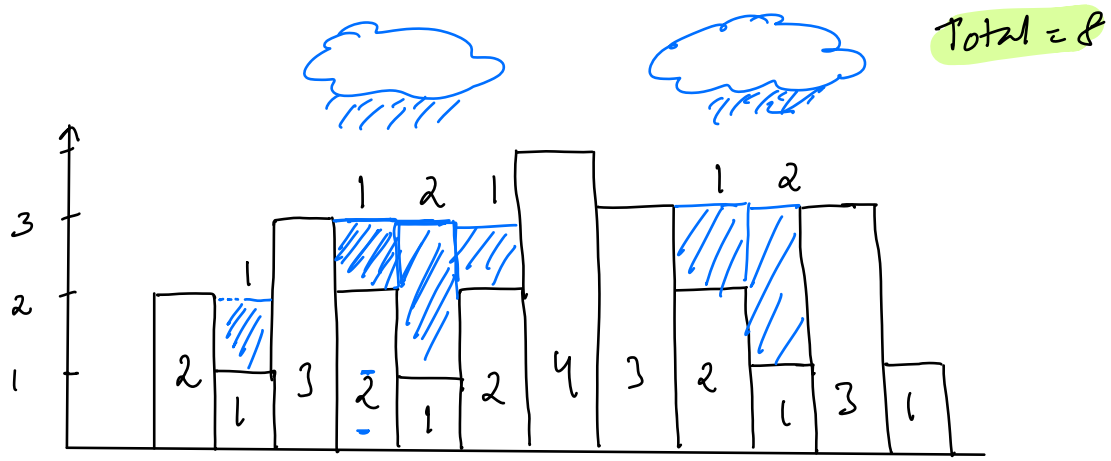
38) Rain watter trapped }

Given N array elements where ar[i] represents height of the building.

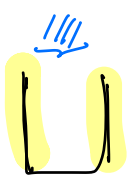
Return amount of water trapped by buildings?

Ex:

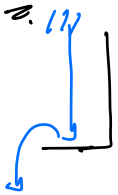
ar[ ] = { 2, 1, 3, 2, 1, 2, 4, 3, 2, 1, 3, 1 }



Ex:

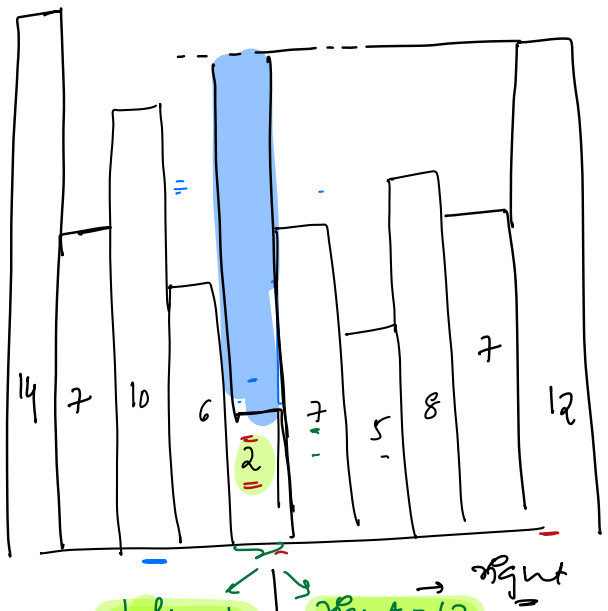


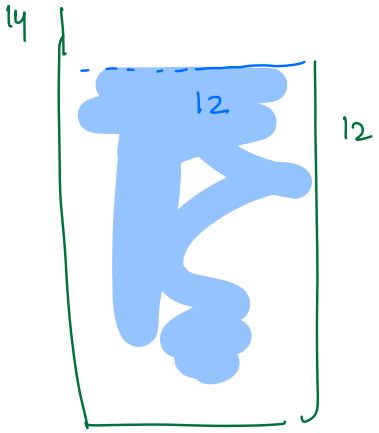
Ex2:



Obs: boundaries will hold water

Eg 1:





left : 14 → left max

right = 12 → right

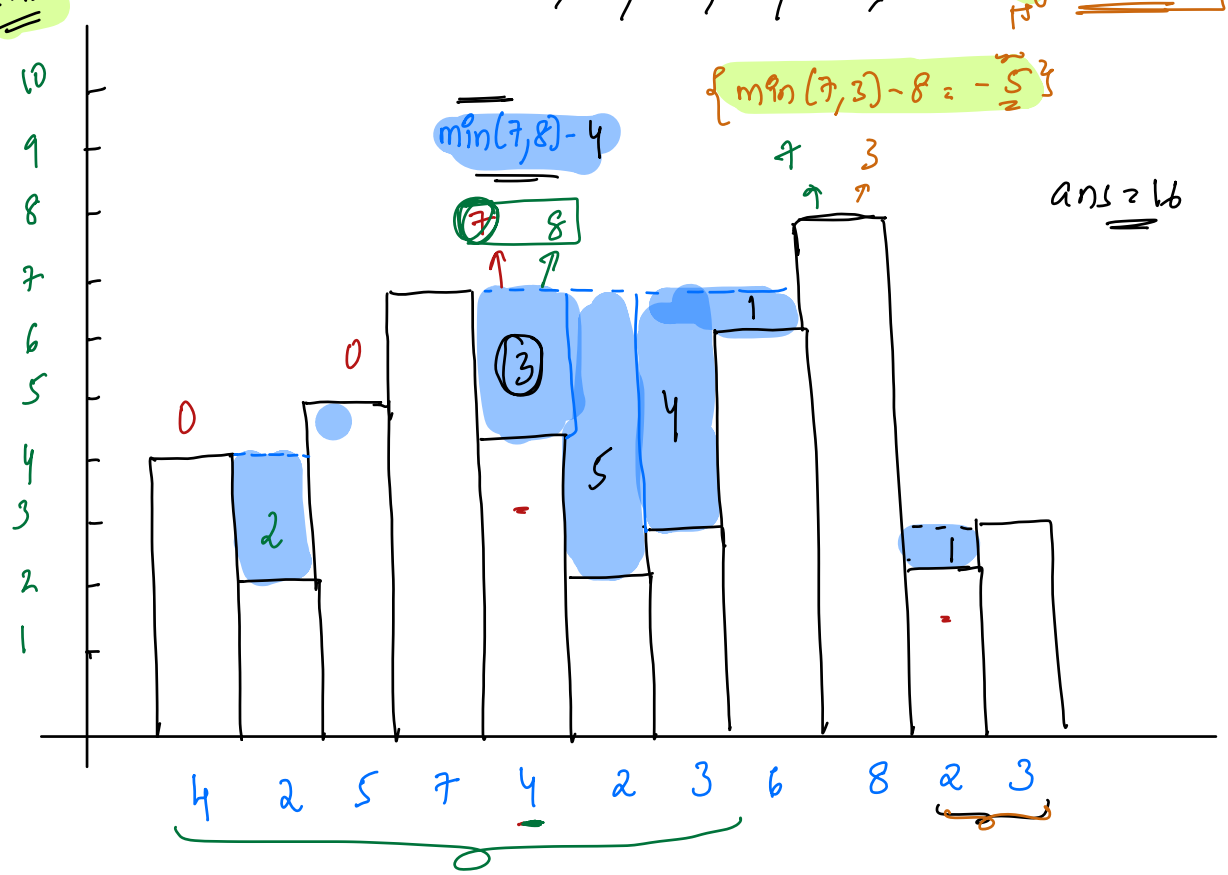
water can : min (14, 12) − 2 = 10

Total water = sum of water accumulated on each bar

Eg 2:

13, 11, 15, 16, 12, 17

any -ve = 0



Sum = 0

for (i = 1; i < N-1; i++) {

// for i^{th} index building

lmax = max [0, i-1] (Pmax[i-1])

Rmax = max [i+1, N-1] (Smax[i+1])

h = min (lmax, Rmax)

w = max (h - ar[i], 0)

Sum = Sum + w

}

- h - ar[i] → -ve
- 0 → 0

i == 0 Edge
→ no first Building
no water, max

i == N-1 Edge
→ no last Building
no water.

Can we optimize
pMax[i] =
max of [0 i]

Can we optim
sMax[i] =
max of [i N-1]

TC: O(N * [N + N]) SC: O(1)

max left max right

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

→ 10:53

48) Min swaps required to bring all elements = x together

consecutive

Ex: x = 8 , len = 5

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

[0 4] = 3 swaps

[1 5] = 2's 8 ⇒ 5 - 2 = 3 swaps

[2 6] = 3's 8 ⇒ 5 - 3 = 2 swaps

[3 7] = 2's 8 = 5 - 2 = 3 swaps

[4 8] = 3's 8 = 5 - 3 = 2 swaps

[5 9] = 3's 8 = 5 - 3 → 2 swaps

final ans = 2

// Idea: Given x, iterate on get count = k

// 1st subarray: [0, k-1] : k - get no: of x's

// 2nd subarr : [1 k] : k - get no: of x's [1 k]

// 3rd sub : [2, k+1] : k - get no: of x's [2 k+1]

⋮

// last sub : [N-k, N-1] : k - get no: of x's [N-k, N-1]

flip means for entire subarray
make 1 → 0 & 0 → 1

508) Flip: find the max no: of 1's which, when flipped will give us max no: of 1's in array.

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

Total 1's

[2 4] :	0	1	0	1	1	1	0	0	1	0	: 5
[3 4] :	0	1	1	1	1	1	0	0	1	0	: 6
[3 7] :	0	1	1	1	1	0	1	1	1	0	: 7

//

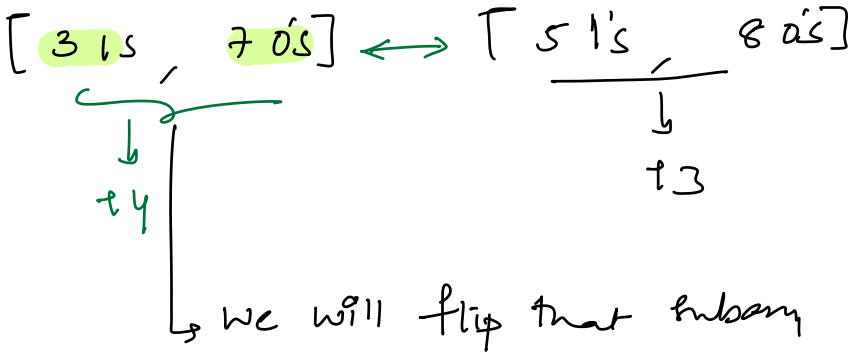
Hint: 1) Flip every subarray & get count : ✓

2) Subarray: [5 1's, 0's 3] | [3 1's, 6 0's]

[5 1's, 0's 3] → -2

[3 1's, 6 0's] → +3

[2 1's, 7 0's] → +5 profit



when nos of 0s - no: of 1s = max

// [1 0 -1 +1 0 1

// Replace 1 → -1 represent loss; 0 → +1 represent profit

0 +1 O(N)

↳ get subarray with max profit

68)

Given N Array Elements, find length of longer subarray which can be re-arranged in a strictly increasing by 1 : {1 ↔ 3}

Ex: 3 6 [1 3 2] 10 4

↳ [1, 2, 3] ✓ len : 3

↑1 ↑1

Ex: 4 [8 6 7 9] 2 1 [5]

[6 7 8 9] →

↑1 ↑1 ↑1

// 2D kadane's

68] Given a matrix → [N * M]

P1: Given a matrix find max submatrix sum where submatrix starts row=0, end row = N-1

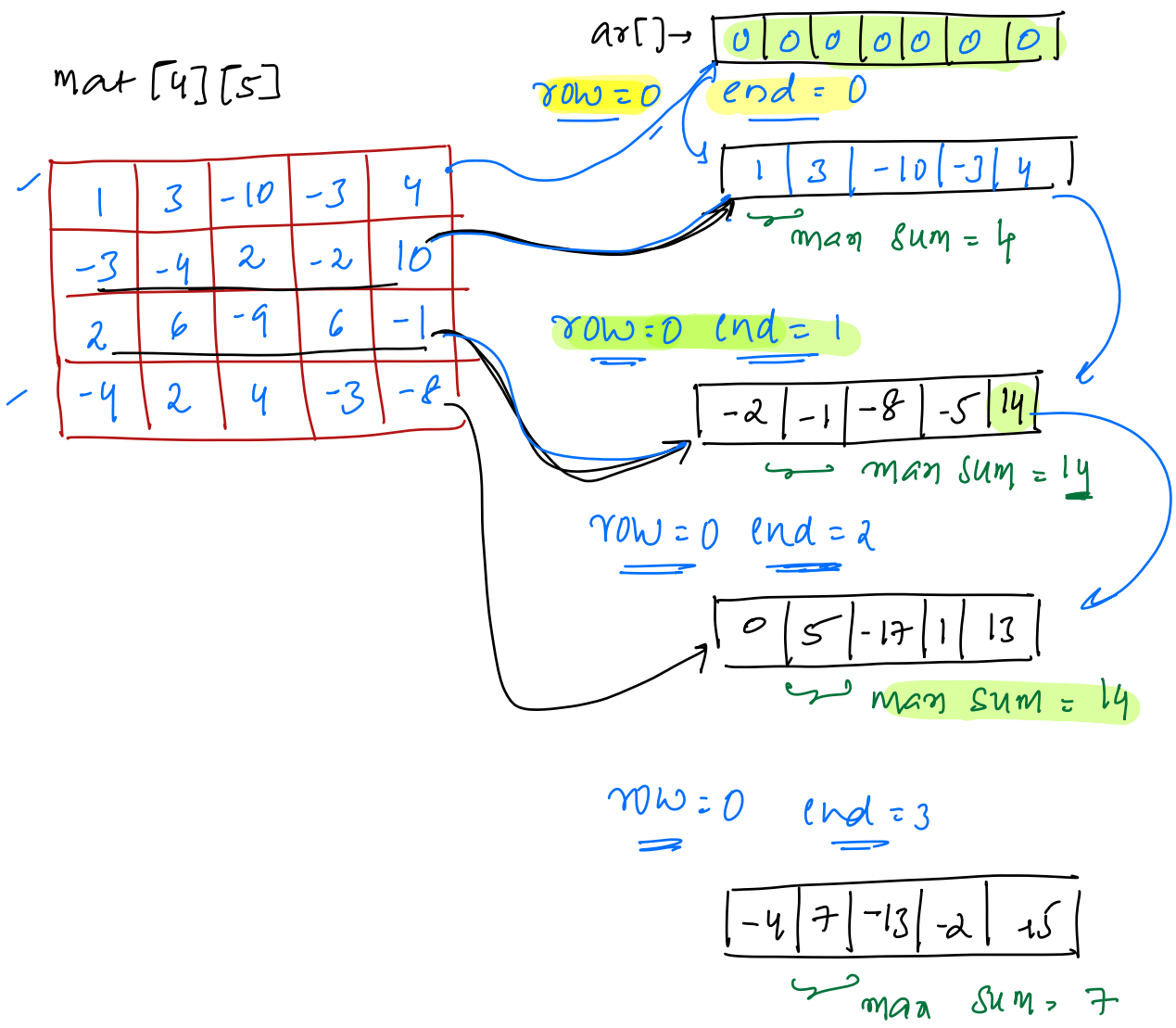
Ex1: mat [3][5] → Submatrix = $m * \frac{(m+1)}{2}$

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

Apply kadane's & get max sum

-13	7	1	9	-4

P2 : Given a matrix ~~find~~ max submatrix sum where submatrix starts row = 0, end anywhere?



// Pseudo Code

```
ans = mat[0][0]
str = 0 ; str < N ; str++) {          -> O(N)

    int ar[m] = {0};
    endr = str ; endr < N ; endr++) {  -> N
        j = 0 ; j < m ; j++) {         -> m
            ar[j] += mat[endr][j]
        }
        val = maxSubSum(ar, m) }       -> m }   (Kadane's logic)
        ans = max(ans, val)
    }
}
```

-> TC: $O(N * N * (m+m))$ SC: $O(m)$

-> $O(N^2 m)$

// merge Intervals :

↳ Sort me code

Debug] doubt Ses