Todays Content:

- max subarray sum ✓
- max submatrix sum 25th Problem Solving
- row_wise column wise sorted max submatrix sum ✓
- find max no of 1's in any row - 3 ✓
- Max absolute diff : 25th problem solving

Max Subarray Sum:

Given ar[N] find max subarray sum
- subarray → { Continous part of ar[] }
- sum → order

Ex1: ar[7] = { 3 2 -6 8 2 -9 4 } ans = 10

(sum = 9 over indices 0 1 2 3 4; sum = 10 over indices 3 4)

Ex2: ar[7] = { -3 2 4 -1 3 -4 3 } ans = 8

(indices 0 1 2 3 4 5 6; sum = 8 over indices 1 2 3 4)

Idea → Sort arr[]: if we sort, it affects array

↳ for every subarray [i, j] : { Revise Subarrays }

: Iterate get sum & get overall max ⇒ TC: $O(N^2 * N)$: $O(N^3)$

: Get sum of subarray [i, j] using pf[] TC: $O(N^2 * 1)$: $O(N^2)$

↳ SC: O(N) ... O(1)

pf[n] size

Optimization:

Case I : If all ar[] > 0

ar[ ] = 4 2 1 6 7

ans = sum of all ar[] elements

Case II: If all ar[] < 0

ar[] = -4 -8 -9 -3 -5

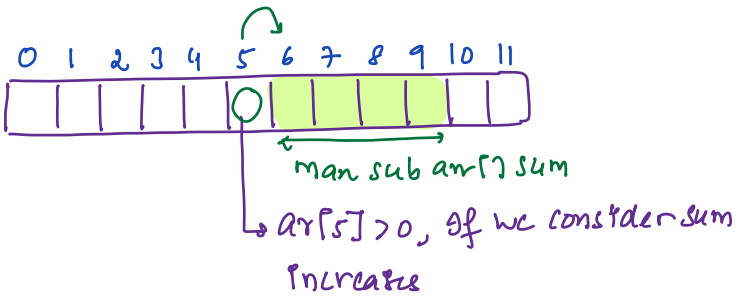
ans = max of ar[] data

Case-III : ar[] :

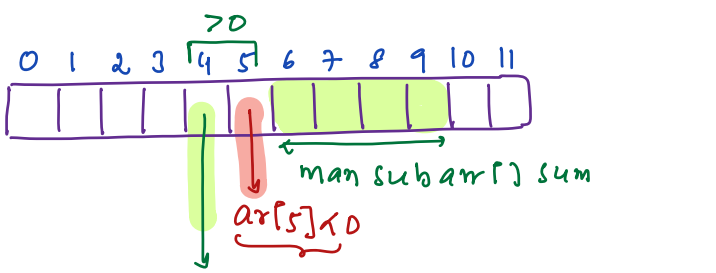
ans = sum of all the elements

If we include elem left overall sum reduce

Case-IV : ar[] :



Case-V : ar[]:



ar[4] > 0, say ar[4] + ar[5] > 0

Idea: We will carry sum, if sum > 0

	0	1	2	3	4	5	6	7	8	9	10	11
ar[] =	5	6	7	-3	2	-10	-12	8	12	21	-4	7

S = 0 → 5 → 11 → 18 → 15 → 17 → 7 → ~~-5~~ 0 → 8 → 20 → 41 → 37 → 44

a = INT_MIN → 5 → 11 → 18 → 18 → 18 → 18 → 18 → 18 → 20 → 41 → 41 → 44

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

S = 0 → ~~-3~~ 0 → 6 → 4 → 8 → 3 → 5 → 6 → 0

ans = INT_MIN → -3 → 6 → 6 → 8 → 8 → 8 → 8 → 8

Kadane's Algo

```
int subSum(int ar[], int n){ TC: O(N) SC: O(1)

    int sum = 0, ans = ar[0]   [ ar[0] is valid subarray sum
                               [ ans = INT_MIN

    i = 0; i < n; i++){
        sum = sum + ar[i]
        if(sum > ans){ ans = sum}
        if(sum < 0){ sum = 0}
    }
    return ans;
}
```

ar[] = { -6 -8 -3 -7 }

Please 2/3 mins ka dry run on your own. →

Sum = 0 → ~~-6~~ 0 → ~~-8~~ 0 → ~~-3~~ 0 → -7

ans = -6 → -6 → -6 → -3 → -3

TODO

Find how to calculate start & end index of subarray with max sum

3Q) Given binary mat[N][M], row-wise sorted find max no: of 1s in a row.

mat[6][7]: a = 2, ans = M - a = 7 - a = 7 - 2 = 5

	0	1	2	3	4	5	6
0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	1
2	0	0	0	0	1	1	1
3	0	0	0	1	1	1	1
4	0	0	1	1	1	1	1
5	0	0	0	0	1	1	1

Idea: 1) Iterate on every row get total 1s & get overall max

TC: O(N*M) SC: O(1)

2) Find the 1st column which contains 1s = a

Initialize a = M // Edge Case

ans = M - a

M → no: of columns

a = 0 *

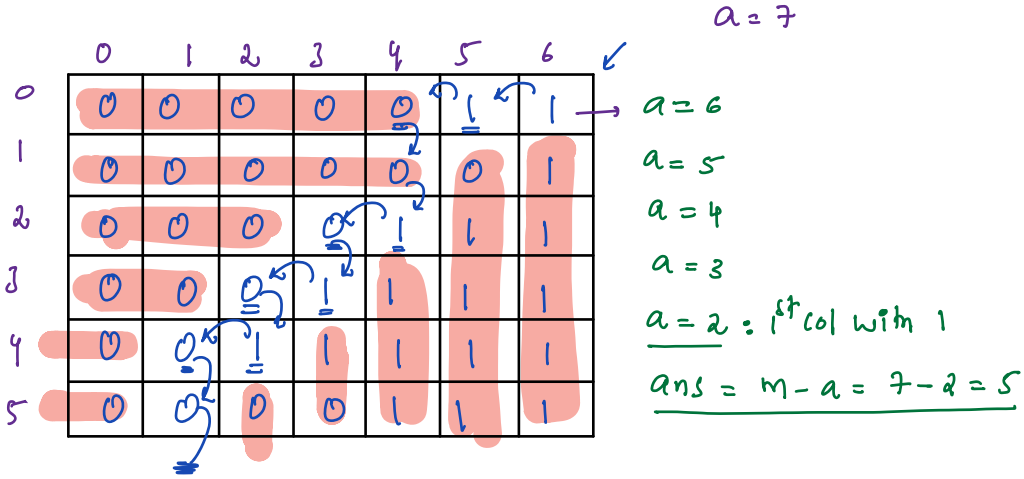
N M

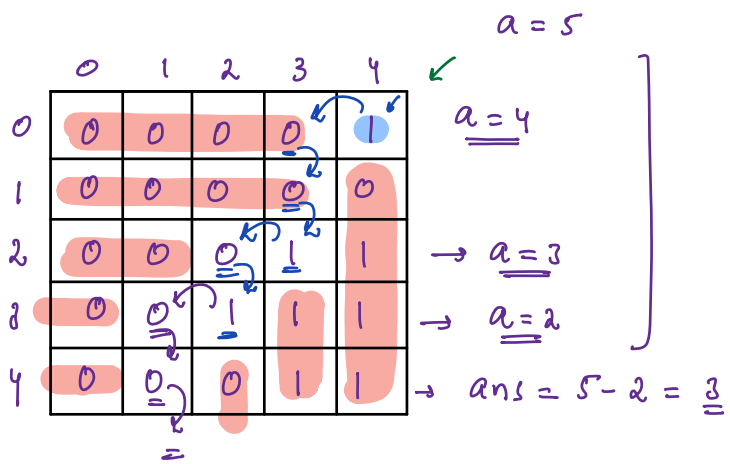
mat[2,5] =

0	0	0	0	0	0
1	0	0	0	0	0

TC: O(N*M) SC: O(1)

Optige calculate of a : // 1st col which contains 1





int maxis(int mat[N][M]) { TC: O(N+M) SC: O(1)

```
i=0, j= M-1, a = M
while ( i<n && j>=0) {
    if( mat[i, j] == 1){
        a=j ; j--
    }
    else{
        i++
    }
}
return m-a
}
```

38) Given row wise column wise sorted matrix, find max submatrix

Ex1:

	0	1	2	3
0	-20	-16	-10	-7
1	-15	-9	4	6
2	-10	3	7	9
3	-3	7	10	12

Sum = 49

Ex2:

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

Sum = 15

Ex3:

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

Sum = -3

Obs: mat[N][M]

ans sub

TL BR

[Top left can be anything in matrix] (N-1, M-1)

```
int maxSum(int mat[N][M]) { TC: O(N*M) SC: O(N*M)
    //Construct pfmat[N][M] TODO → O(N*M)
    ans = mat[0,0]
    for(i=0; i<N; i++) {
        for(j=0; j<M; j++) {        } → O(N*M*i) →
            TL = (i, j), BR = (N-1, M-1)
            Submatrix found, get it's sum
            using pfmat[][], say it's = v
            ans = max(ans, v)
        }
    }
    return ans;
}
```