Todays Content:

- Max submatrix sum
- Find k in 2D matrix
- Beggers outside temple
- Merge Intervals
    a) Merge all overlapping intervals
    b) Insert a interval into non-overlapping interval

Q) Given row wise column wise sorted matrix, find max submatrix sum

Eg1:

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

Sum = 152

Eg2:

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

Sum = 15

Eg3:

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

Sum = -3

Ideas:

1) For every submatrix, get the sum, & find overall max

TC: $O(N^2M^2) * O(N*M) = O(N^3M^3)$

// Can be optimized using pfm[][]

TC: $O(N*M) + O(N^2M^2) * O(1)$

SC: $O(N*M)$

// All submatrices

// Calculating sum of each sub matrix using pf[][]

//obs: Our ans submatrix bottom right = [n-1, m-1] cell

↳ Take every cell as TL

Create pfm[][]

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

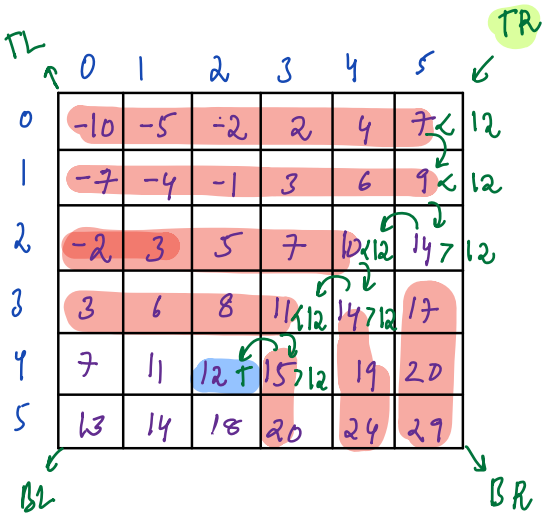
SC: $O(N*M)$

```
i = 0; i < N; i++) {
    j = 0; j < M; j++) {
        TL = (i, j)   BR = (n-1, m-1)
        Using pfm[][] get submatrix sum
        & overall max
    }
}
```

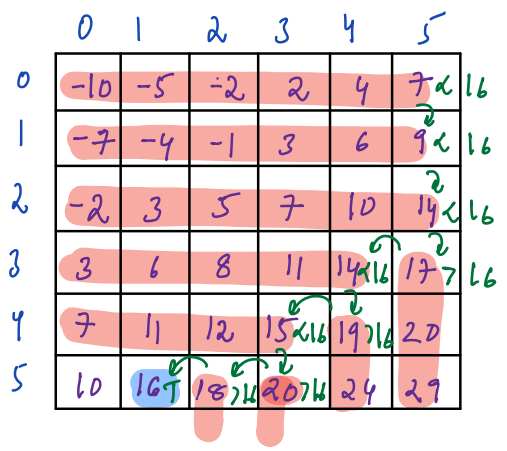
28) Given row wise column wise sorted matrix, find k ?

return True / False

k = 12



k = 16



TODO: TL, BL, BR, TR ✓

row wise column
Both inf helped us

Idea:

1) Iterate on entire matrix
   TC: O(N*M) SC: O(1)
2) Iterate on only those rows, which can contain k
   TC: O(N*M) SC: O(1)

Pseudocode

```
i = 0, j = m-1
while (i < n && j >= 0){
    if(mat[i,j] == k){
        return true
    }
    if(mat[i,j] < k){
        // skip row
        i++
    }
    else // skip colum
        j--
}
return false
```

TC: O(N+M): At every step, we either skip a row or column

- Total Rows we can skip : N
- Total cols we can skip : M

Total skips at max = N+M

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

38) Given a ar[N], s.t ar[] = 0, initially return the final array after performing all the queries

Queries → (i, x) = Add x to all the numbers from A[i] to A[N-1]

Eg1:

	0	1	2	3	4	5	6
A =	0	0	0	0	0	0	0
		3	3	3	3	3	3
					-2	-2	-2
				1	1	1	1
	0	3	3	4	2	2	2

(i, x) : 3 Queries

(1, 3)
(4, -2)
(3, 1)

Idea1: For every query iterate from [i, N-1] and add x

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

Eg1:

	0	1	2	3	4	5	6
A =	0	0	0	0	0	0	0
Mod A =	0	3	0	1	-2	0	0
Pf A[]	0	3	3	4	2	2	2

(i, x) : 3 Queries

(1, 3)
(4, -2)
(3, 1)

Idea2: TC: O(Q + N)

1) For every query (i, x) ar[i] += x
2) After all queries apply Pf[] on ar[]

Queries → (i, j, x) = Add x to all numbers from A[i] → A[j]

A[] = [0 0 0 0 0 0 0]

2 2 2

3 3 3 3

-1 -1 -1

A[]: 0 2 4 4 2 3 0

i j x : 3

(1 3 2)

(2 5 3)

(2 4 -1)

Ideal: For every query iterate & add x

TC: O(Q*N)

0 1 2 .. i i+1 .. j j+1 j+2 .. N-1

(i j x):

x x ... x x x ... x : Q(i, x)

-x -x ... -x : Q(j+1, -x)

→ (i, x) + (j+1, -x)

ar[i] += x ar[j+1] -= x

A[] = [0 0 0 0 0 0 0]

mod A[] 0 2 2 0 -2 1 -3

Pf A[]: 0 2 4 4 2 3 0

i j x : 3

(1 3 2)

(2 5 3)

(2 4 -1)

Pseudocode: → TC: O(Q + N)

1) For all Queries

Q: (i, j, x):

ar[i] += x

if (j+1 < n) {

ar[j+1] -= x // If j = n-1, j+1 = n, out of bounds

}

8:27am → 8:37am

2mins Queries

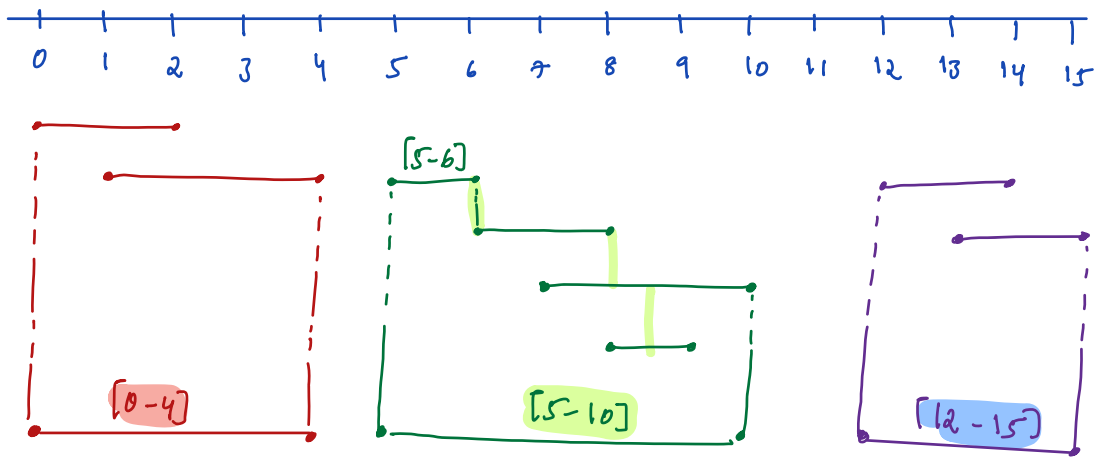
2) Apply pf[] on ar[]

4Q) Given a sorted list of overlapping intervals, sorted based on start time, merge all overlapping intervals & return a sorted list of non-overlapping intervals : [result-list]

[s e]

Intervals:

	0	1	2	3	4	5	6	7
s	[0	1	5	6	7	8	12	13]
e	[2	4	6	8	10	9	14	15]



Final Intervals: [Non Overlapping Intervals, Sorted on start time]

[[0 4]
[5 10]
[12 15]]

s	e	→	x	y
[0	2] ✓		[0	2]
[1	4] ✓		[0,	4]
[5	6]	update x, y →	[5,	6]
[6	8] ✓		[5,	8]
[7	10] ✓		[5,	10]
[8	9] ✓		[5,	10]
[12	14]	update x, y →	[12,	14]
[13	15]		[12,	15]

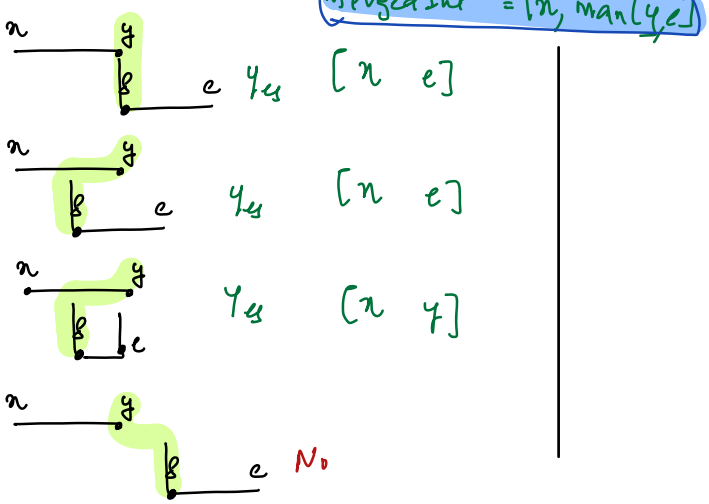
[Take care of last interval]

Final ans

[0, 4]
[5, 10]
[12, 15]

non-overlapping intervals sorted based on start time
TC: O(N)

MergedInt = [x, max(y, e)]




```
void merge(int s[], int e[]) {   TC: O(N)
    int x = s[0], y = e[0]
    for(i=1; i<n; i++) {
        // ith interval is overlapping x — y
        if(y >= s[i]) {  // merged interval
            y = max(y, e[i])
        }
        else {  // non-overlapping
            print(x, y)
            x = s[i], y = e[i]
        }
    }
    print(x, y)
}
```

Q) Given N non_overlapping intervals, sorted based on starttime,

Insert new interval & return non_overlapping intervals

[s e]

N=8

new Interval

[1 3]

[4 7]

[10 14] → [12 22] : {10, 22}

[16 19] → {10, 22} : {10, 22}

[21 24] → {10, 22} : {10, 24}

[27 30] * {10, 24}

[32 35]

[36 40]

[1 3]
[4 7]
} Intervals before new interval

{10, 24} } Intervals merging new interval

[27 30]
[32 35]
[36 40]
} Intervals coming after new interval

N=5:

new Interval

[1 5]

[8 10]

[11 14] → [12, 22] : {11, 22}

[15 20] → {11, 22} : {11, 22}

[20 24] → {11, 22} : {11 24}

[1 5]

[8 10]

{11 24}

TODO: TC: O(N)

1) [s e] coming before [x y]
2) [s e] coming after [x y]
3) [s e] merging [x y]