Agenda:

1. Find Maximum Subarray Sum
2. Find in row-wise and col-wise sorted matrix
3. Merge sorted Overlapping Intervals
4. Transpose of a square matrix
5. Rotate a matrix to 90 degree clockwise

Q → Given an integer array find max subarray sum.

continuous part of array

```
          0   1   2   3   4   5   6
A = [ -20  30   5  16  -2   0   8 ]
```

Ans = 57

```
          0   1   2   3   4
A = [  4   5   2   1   6 ]
```

Ans = 18

```
          0   1   2   3   4
A = [ -4  -3  -6  -9  -2 ]
```

Ans = -2

Bruteforce → ∀ subarrays calculate sum & take max.

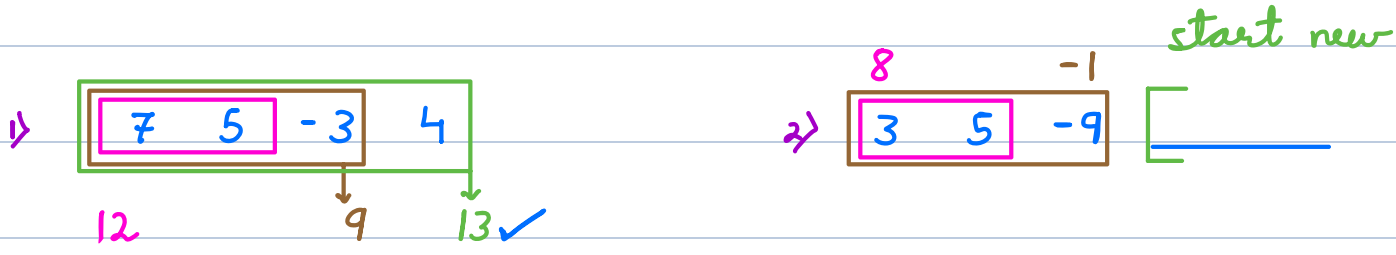
TC = $O(N^3)$ SC = $O(1)$

Carry Forward →

```
ans = A[0]
for i → 0 to (N-1) {
    sum = 0
    for j → i to (N-1) {
        sum += A[j]
        ans = max(ans, sum)
    }
} return ans
```

TC = $O(N^2)$ SC = $O(1)$

Kadane's Algorithm



	0	1	2	3	4	5	6	7	8	9
A = [	5	2	-3	-8	6	1	-5	11	0	-3]
sum =	~~5~~	~~7~~	~~4~~	~~-4~~ → 0	~~6~~	~~7~~	~~2~~	~~13~~	~~13~~	10
ans =	~~5~~	~~7~~	13							

0 1 2 3 4 5 6

A = [-2 3 4 -1 5 -10 7]

sum = -2 → 0, 3, 7, 6, 11, 1, 8

ans = ~~-2~~ ~~3~~ ~~7~~ 11

ans = A[0]

sum = 0

0 1 2

A = [-5 -3 -8]

sum = -5 → 0, -3 → 0, -8 → 0 ans = ~~-5~~ -3

```
for i → 0 to (N-1) {
    sum += A[i]
    if ( sum > ans ) ans = sum
    if ( sum < 0 )   sum = 0
}
return ans
```

TC = O(N) SC = O(1)

---

ans = A[0] L = 0 R = 0 // L — R

sum = 0 st = 0

```
for i → 0 to (N-1) {
    sum += A[i]
    if ( sum > ans ) { ans = sum  L = st  R = i }
    if ( sum < 0 ) { sum = 0  st = i+1 }
}
return ans
```

Q → Given a row-wise col-wise sorted matrix. (Ascending)

Check if given element K is present.

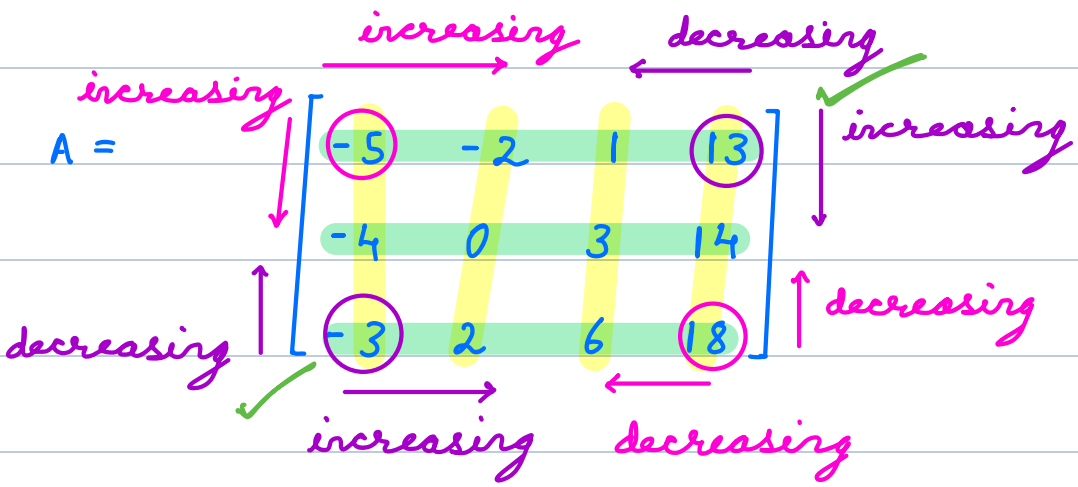
$$A = \begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 6 & 18 \end{bmatrix}$$

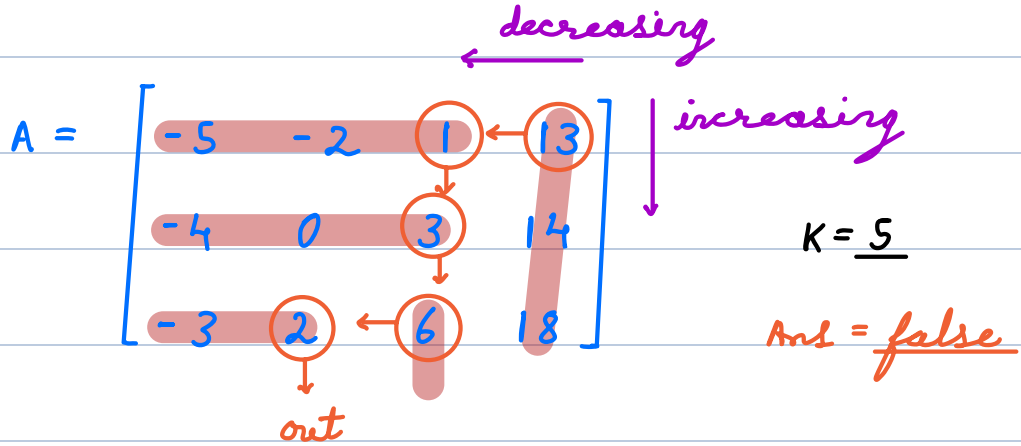
K = 0 Ans = True

K = -1 Ans = false

Bruteforce → ∀ i, j check if A[i][j] == K.

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





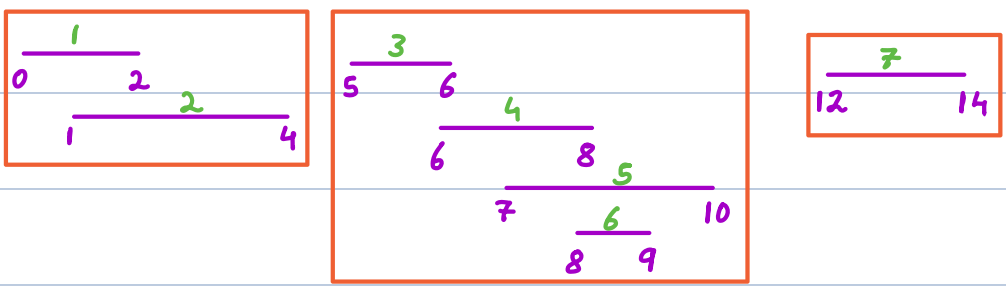
K = 5

Ans = false

```
i = 0      j = M-1
while ( i < N  &&  j >= 0 ) {
    if ( A[i][j] == K )  return true
    if ( A[i][j] < K )   i++
    else                 j--
}
return false
```

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

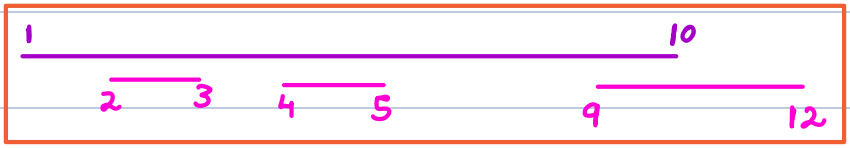
Q → Given a list of intervals, sorted wrt start time. Merge all overlapping intervals & return the sorted list of non-overlapping intervals.



Ans → 0 — 4, 5 — 10, 12 — 14

Slot A & B are overlapping if **start of B <= end of A.**

A = [(1, 10) (2, 3) (4, 5) (9, 12)]



S = A[0][0] E = A[0][1] // A[i][0] —— A[i][1]

```
for i → 1 to (N-1) {
    if ( A[i][0] <= E )      E = max ( E, A[i][1] )
    else {   print ( S, E )
                 S = A[i][0]      E = A[i][1]
             }
}
print ( S, E )
```

TC = O(N) SC = O(1)

	i	i	i	i	i	i
	0	1	2	3	4	5	6
A =	0	1	5	6	7	8	12
	2	4	6	8	10	9	14

S = ~~0~~ ~~5~~ 12

E = ~~2~~ ~~4~~ ~~6~~ ~~8~~ ~~10~~ 14

o/p → 0, 4

5, 10

12, 14

Q → convert the given square matrix to its transpose.

row ↔ column

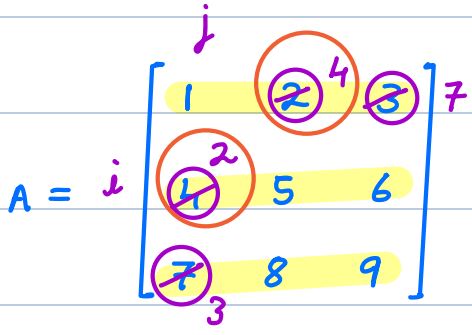
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Matrix Transpose → 7 min

$i, j \longleftrightarrow j, i$

```
for i → 0 to (N-1) {
    for j → i+1 to (N-1) {
        t = A[i][j]          // swap i,j ↔ j,i
        A[i][j] = A[j][i]
        A[j][i] = t
    }
}
```

TC = O(N²) SC = O(1)



Q → Rotate the given sq. matrix by 90° clockwise.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Transpose

90° rotated

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Reverse every row

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Sol → 1) Transpose the given matrix

2) Reverse every row

TC = $O(N^2)$ SC = $O(1)$