7:05 am

{ Answers → Me }
{ Qns → Everyone }

Q1: Given a[N]. Print the max subarray sum of length = K

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

K = 5

S	e	Sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

ans = 16

idea 1:
For every subarr of len = K, iterate & get its sum & calc overall max.

```
int subSum(int a[], int N, int K){
    s = 0
    e = K-1,   ans = INT_MIN      ↦ find in own lang
    while(e < n){
        sum = 0
        for(i=s; i<=e; i++){
            sum += a[i]
        }
        if(sum > ans){
            ans = sum
        }
        s++, e++;
    }
    return ans
}
```

(*) [s e] = e - s + 1

[0 e] = e - 0 + 1 = K

e = K-1

(*) [s n-1] = K

n - 1 - s + 1 = K

s = n - K

TC: O(Total no. of subarr of len = K * iters per subarr)

⇒ O((n-K+1) * K)

= O(n²)

SC: O(1)

① [0 k-1]
② [1 k]
③ [2 k+1]
⋮ ⋮
?? [n-k n-1]

No of terms = No of subarrs

0 → 1 → 2 .... → n-k

[0 n-k]

= n-k+1

O((n-k+1) * k)

k=1

O((n-1+1)×1)

= O(n)

k=n

O((n-n+1) n)

= O(n)

k = n/2

↳ O((n - n/2 + 1) n/2)

⇒ O((n/2 + 1)(n/2))

$= \frac{n^2}{4} + \frac{n}{2}$

$= O(n^2)$

idea 2: Use prefix sum

↳ for every subarr of len=k, get its sum using pf[]

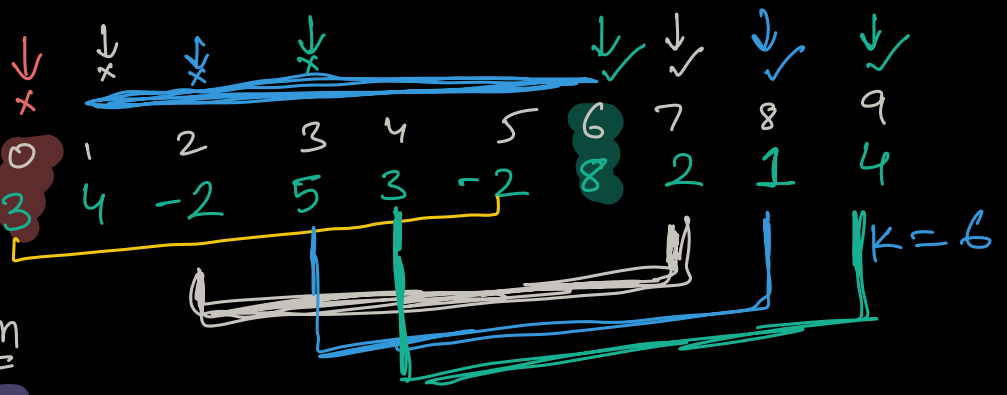
& calc sum & find max

TC: O(N + (N-K+1)×1) = O(N)

SC: O(N) ↳ pf[]

To-do:

idea 3:



s	e	sum
0	5	5
1	6	Sum = Sum - a[0] + a[6] = 5 - (-3) + 8 = 16
2	7	Sum = Sum - a[1] + a[7] = 16 - 4 + 2 = 14
3	8	Sum = Sum - a[2] + a[8] = 14 - (-2) + 1 = 17 ✓ ← ans.
4	9	Sum = Sum - a[3] + a[9] = 17 - 5 + 4 = 16

↔ window

↳ Carry forward & subarr size = same

↳ sliding window

Pseudo code:

```
int SubsumK (int a[], int N, int K) {
    sum = 0
    for(i=0; i<K; i++) {          // first window sum
        sum += a[i]               // [0  K-1]
    }                             //  ↓   ↓
                                  // [i   K]
    ans = sum
    s = 1 , e = K
    while(e < n) {
        sum = sum - a[s-1] + a[e]
        ans = max(ans, sum)
        s++, e++
    }
    return ans
}
```

K [ ]

n-K [ ]

TC: $O(K+n-K) = O(n)$

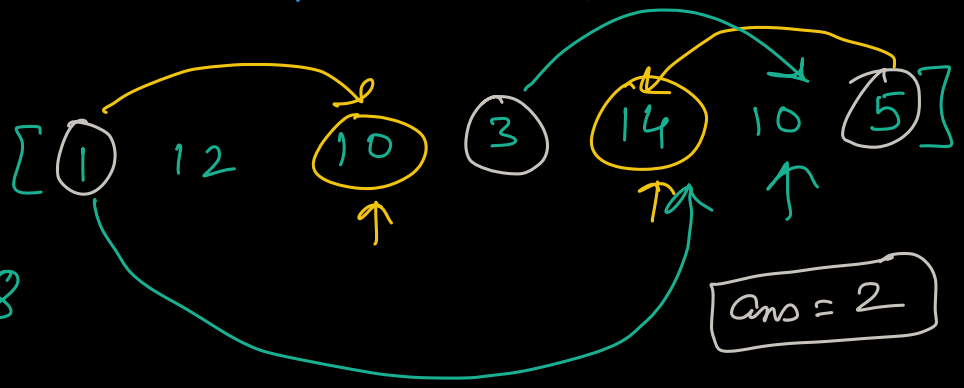
SC: $O(1)$

Q2: Given a[N] & a no. B. Find & return min. no. of swaps to bring all numbers <= B together.

→ swap any 2 numbers

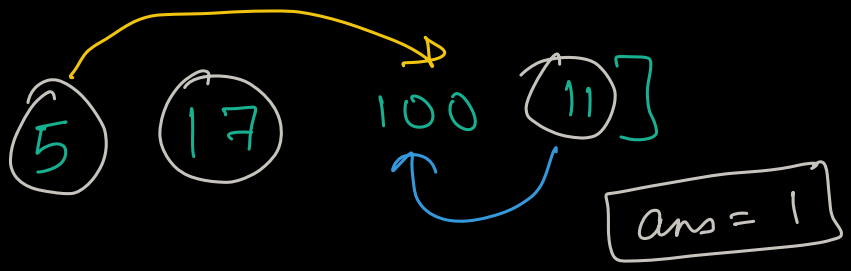
Eg: A = [1, 12, 10, 3, 14, 10, 5]

B = 8



A = [5, 17, 100, 11]

B = 20



#obs1: all no. <= B ⇒ together.

↳ window size = count of elements <= B

A = [8, 3, 10, 20, 22, 13, 1, 2, 55, 5, 15, 50]

(indices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)

B = 5

window size = 4.

s	e	# of swaps
0	3	3
1	4	3
2	5	4
3	6	3
4	7	2

s	e	# of swaps
5	8	2
6	9	1
7	10	2
8	11	3

* Count no. of elements <= B = count

* Sliding window of size = count

  ↳ for every window, count of bad = no. of swaps

  ↓

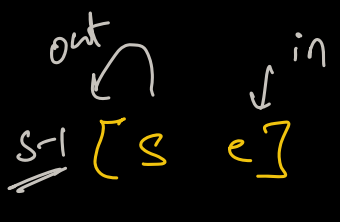
  find min

```
int solve(int a[], int N, int B) {
    cnt = 0
    for(i=0; i<N; i++) {
        if(A[i] <= B) {
            cnt++
        }
    }
    if(cnt <= 1) {
        return 0
    }
    bad = 0
    for(i=0; i<cnt; i++) {
        if(A[i] > B) {
            bad++
        }
    }
    ans = bad
    s = 1, e = cnt
    while(e < N) {
        if(A[s-1] > B) { bad-- } // outgoing
        if(A[e] > B) { bad++ } // incoming
        ans = min(ans, bad)
        s++, e++
    }
```

// Count of good elements

[(2), 3, 4, 5, 6]
B = 2
↳

// get no. of bad in first window

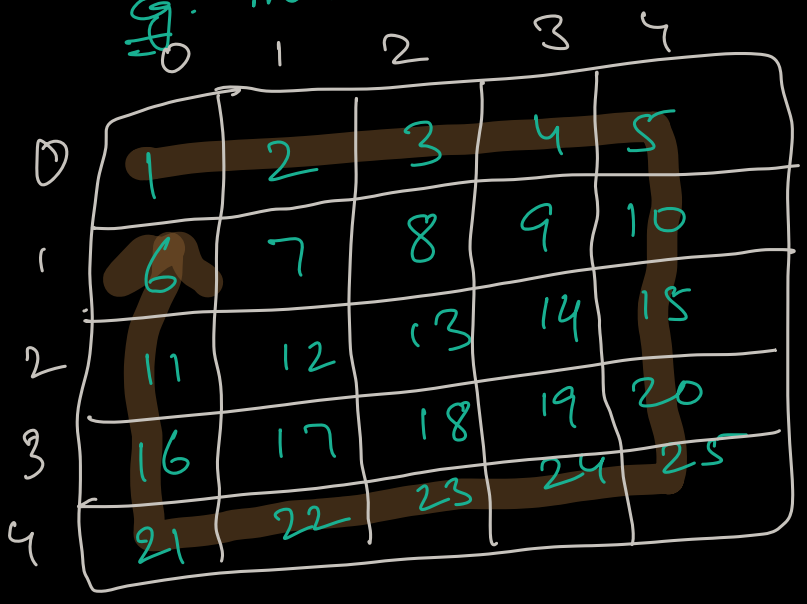


```
    }
}
return ans
}
```

Break → 9:00am

Q3: Given a mat[N][N]. Print the boundary in 2 dir.

Eg: mat[5][5]

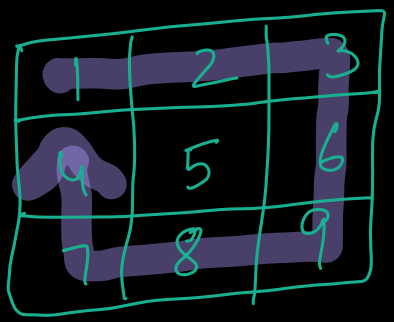


output

1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6

in one single line

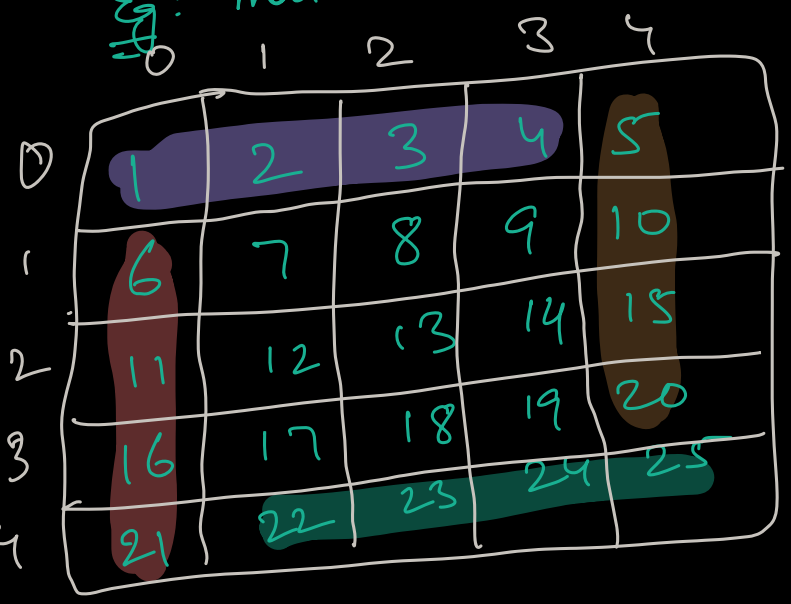
Eg: mat[3][3]



output:

1 2 3 6 9 8 7 4

Eg: mat [5][5]



5x5

4 iter →

4 iter ↓

4 iter ←

4 iter ↑

idea:

N×N mat

N-1 →

N-1 ↓

N-1 ←

N-1 ↑

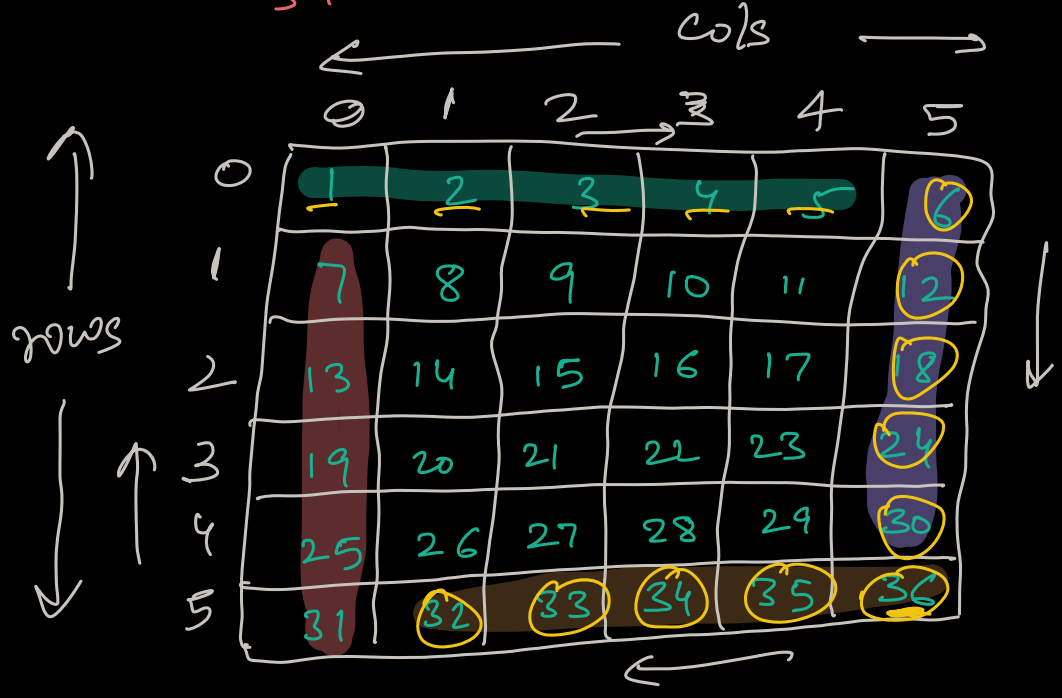
if N=6

↳ 5 iter →

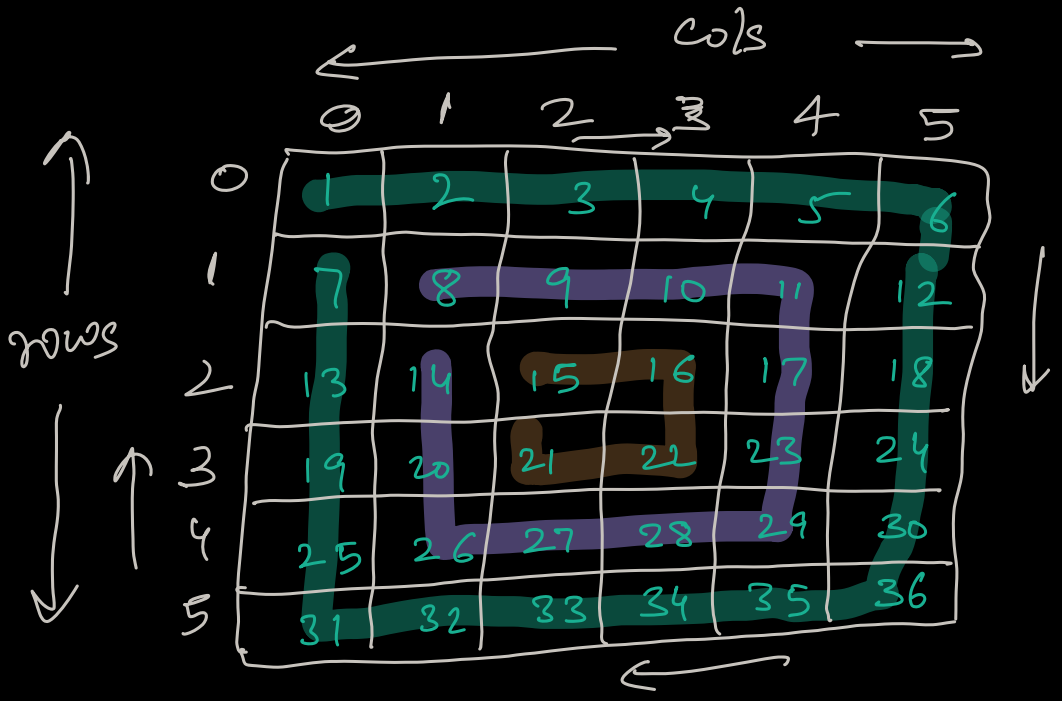
5 iter ↓

5 iter ←

5 iter ↑



```
void printBoundary(int mat[][]) {
    n = mat.length
    row = 0, col = 0  // idx
    // top row : Left to Right
    for(k=1; k<=N; k++) {  // [1 N-1] ⇒ N-1 iter
        print(mat[row][col])
        col++
    }
    // Last col : top to bottom
    for(k=1; k<=N; k++) {
        print(mat[row][col])
        row++
    }
    // Last row: Right to Left
    for(k=1; k<=N; k++) {
        print(mat[row][col])
        col--
    }
    // First col : Bottom to Top
    for(k=1; k<=N; k++) {
        print(mat[row][col])
        row--
    }
}
```



6x6

row	col	N
0	0	6
1	1	4
2	2	2
3	3	0

5 iter

5x5

```
void Spiral(int mat[][]) {
    n = mat.length
    row = 0, col = 0
    while(n > 1) {
        for(k=1; k<N; k++) {
            print(mat[row][col])
            col++
        }
        for(k=1; k<N; k++) {
            print(mat[row][col])
            row++
        }
        for(k=1; k<N; k++) {
            print(mat[row][col])
            col--
        }
        for(k=1; k<N; k++) {
            print(mat[row][col])
            row--
        }
        row++, col++
        n = n-2
    }
    if(n==1) {
        print(mat[row][col])
    }
}
```

TC: $O(N^2)$

SC: $O(1)$