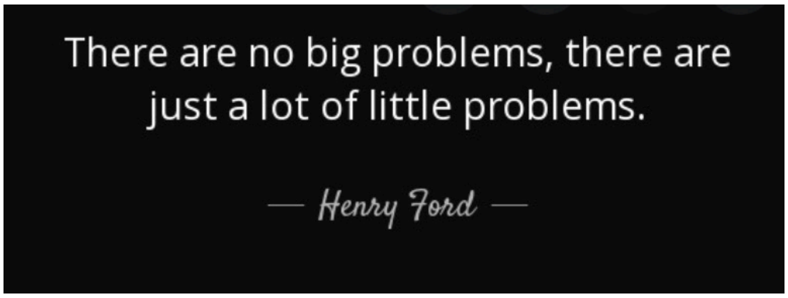
Good Morning Everyone !! ☺



There are no big problems, there are just a lot of little problems.

— Henry Ford —

Today's Content.

- Sliding Window
- 2 problems on 2-D Arrays.

arr =

5	6	2	9	11	-3	5	3	8
0	1	2	3	4	5	6	7	8

	K=1	K=2.	K=3	K.
total no. of sub-arrays of size K	9	8	7	
	[N]	[N-1]	[N-2]	[N-K+1]

Q) Given N elements, print max subarray sum of len = K.

arr[10] :	{	-3	4	-2	5	3	-2	8	2	-1	4	} , $\boxed{K=5}$
		0	1	2	3	4	5	6	7	8	9	

s	e	sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

$\boxed{ans = 16}$

idea1: For every subarray of size K, iterate & calculate sum. Over all max sum will be our ans.

```
int maxSubarray( arr, N, K){
    s=0 , e= K-1 , ans → min integer value.
    while( e < N){
        //iterate & calculate sum
        int sum=0
        for( i = s ; i <= e ; i++){
            sum += arr[i]
        }
        if (sum > ans){ans= sum}
        s ++ , e++ ;
    }
    → return ans.
}
```

T.C → $(N-K+1).K$

$K=1$: $(N-\cancel{K}+\cancel{1}).1$; TC → $O(N)$

$K=N$: $(\cancel{N}-\cancel{N}+1).N$; $O(N)$

$K=N/2$: $(N-\frac{N}{2}+1).(\frac{N}{2}) \approx \frac{N}{2}.\frac{N}{2} = \frac{N^2}{4}$; $O(N^2)$

$\boxed{T.C \rightarrow O(N^2) \quad , \quad S.C \rightarrow O(1)}$

idea-2.

```
①    // Create pSum[N]

②    s = 0 ,  e = K-1 ,  ans → min integer value.

     while( e < N){ // calculating sum of subarray from [s,e]
         sum = 0
         if (s = = 0)  sum = pSum[e]
            else       sum = pSum[e] - pSum[s-1]

         if (sum > ans){ans = sum}
         s++ , e++ ;
     }
  → return ans.
```

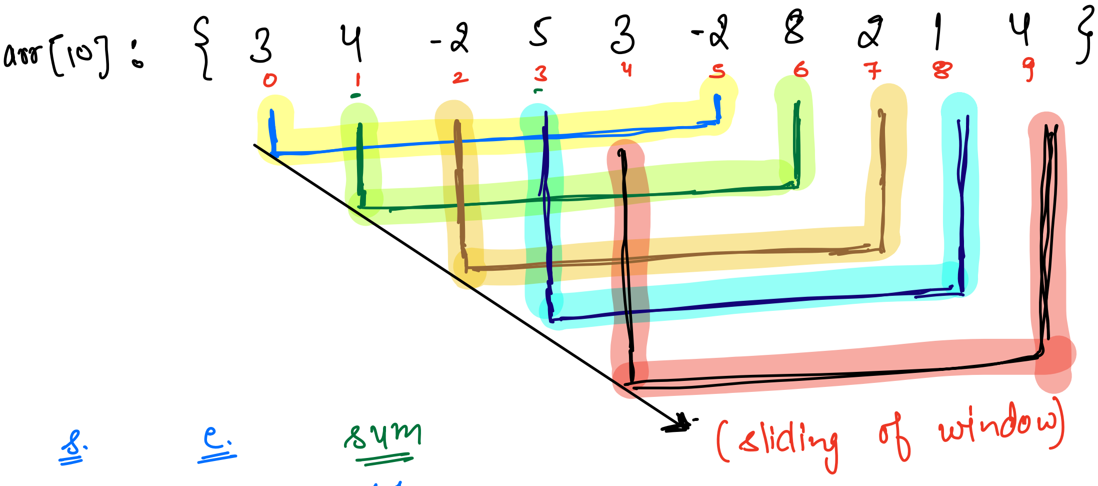
T.C → O(N)
S.C → O(N)

idea-3 :

K=6

arr[10] : { 3 4 -2 5 3 -2 8 2 1 4 }

N=10



s.	e.	sum
0	5	11
1	6	sum = sum - arr[0] + arr[6] = 11 - (3) + 8 = 16
2	7	sum = sum - arr[1] + arr[7] = 16 - 4 + 2 = 14
3	8	sum = sum - arr[2] + arr[8] = 14 - (-2) + 1 = 17
4	9	sum = sum - arr[3] + arr[9] = 17 - 5 + 4 = 16

ans = 17

s e sum = sum - arr(s-1] + arr[e]

[carry forward + All subarrays of same-size ⟶ Sliding Window]

Final code :

```
int  maxSum( arr, N , K) {
    //1. Calculate the sum of first K elements [first window]
        sum = 0
        for( i = 0 ; i < K ; i++) {          ]  K - iteration
            sum += arr[i]                    ]
        }
        ans = sum , s = 1 , e = k
        while ( e < n) {                     ]
            // calculate sum of subarray [s,e].  ]  N-K
            sum = sum- arr(s-1] + arr[e]     ]  iterations
            if (sum > ans) { ans = sum }     ]
            si ++ , ei++ ;                   ]
        }
        return ans.
}
```

T.C → O(N)

S.C → O(1)

Q2) Given arr[N] and a number B. Find and return minimum no. of swaps to bring all numbers <= B together

eg: arr = { $\boxed{1}$ 12 10 $\boxed{3}$ 14 10 $\boxed{5}$ } , B = 8

arr	$\boxed{1}$	12	10	$\boxed{3}$	14	10	$\boxed{5}$
	0	1	2	3	4	5	6

[ans = 2.]

Q arr = { 19 11 3 9 7 25 6 20 4 } , B = 10

arr	19	11	3	9	7	25	6	20	4
	0	1	2	3	4	5	6	7	8

[ans = 1]

Q arr : { 25 30 2 18 7 6 9 3 50 }, B = 10

arr	25	30	2	18	7	6	9	3	50
	0	1	2	3	4	5	6	7	8

[ans = 1]

→ count of all element ≤ B [K]

→ size of sub-array will be fixed. ⇒ K.

→ Find sub-array for which no. of swaps are minimum.

	no. of swaps.
0 – 4	3
1 – 5	2
2 – 6	1
3 – 7	2
4 – 8	1

[ans = 1]

For all elements > B ⇒ **bad elements**

For all elements ≤ B ⇒ Good elements.

pseudo-code.

```
int minSwaps ( arr, N, B) {
    // count no's < B
    K = 0
    for ( i = 0 ; i < N ; i++) {
        if (arr[i] ≤ B) { K++ }
    }
    if ( K= =0 || K= =1) { return 0 }

    // Calculate no. of bad elements for first window.
    bad = 0
    for ( i = 0 ; i < K ; i++) {
        if (arr(i] > B) { bad++ }
    }

    // Apply sliding window technique

        ans = bad , s = 1 , e = k
        while ( e < n ) {
            if ( arr[s-1] > B) bad --      removing arr[s-1]
            if ( arr[e] > B) bad ++        adding arr[e]
            if ( bad < ans) { ans = bad }

            s++ , e++ ;
        }
}
```

T.C → O(N)
S.C → O(1)

Q) Given mat[N][N], print boundary in clockwise direction.

mat[5][5]

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

mat[3][3]

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

O/p → { 1,2,3, 6, 9, 8, 7, 4 }.

O/p. → { 1,2,3,4,5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6 }

Idea:

N-1 →

N-1 ↓

N-1 ←

N-1 ↑

pseudo-code.

```
void    printBoundaryElements ( arr, N ){
    i = 0 ,  j = 0
    //1. print (N-1) elements from l→r

        for ( k = 1 ;  k < N ;  k++){
                print( arr[i][j] ; -
                j ++
        }

    //2. print (N-1) elements from t to d.

        for ( k = 1 ;  k < N ;  k++){
                print( arr[i][j] ;
                i ++
        }

    //3. print (N-1) elements from r to l
        for ( k = 1 ;  k < N ;  k++){
                print( arr[i][j] ;
                j --
        }

    //4. print (N-1) elements from d to t
        for ( k = 1 ;  k < N ;  k++){
                print( arr[i][j] ; -
                i --
        }
}
```

k	i	j
1	0	0
2	0	1
3	0	2
4	0	3
	0	4

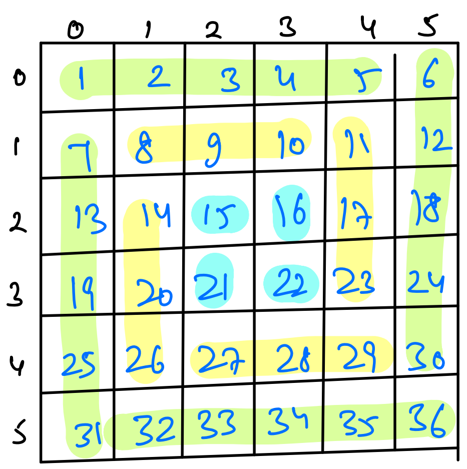
4 4

4 0

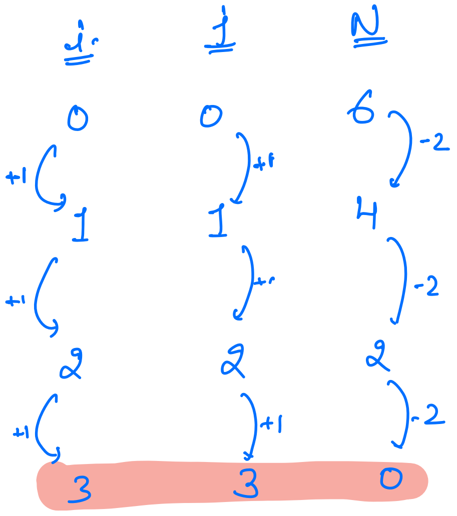
0, 0

$$\boxed{\begin{array}{l} T.C \rightarrow O(N) \\ S.C \rightarrow O(1) \end{array}}$$

arr[6][6]



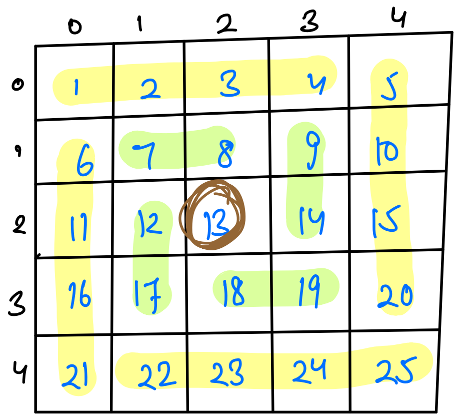
	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36



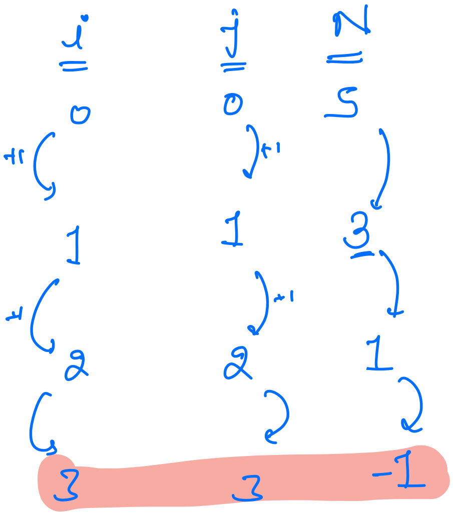
i	j	N
0	0	6
1	1	4
2	2	2
3	3	0

Spiral Printing.

arr[5][5]



	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25



i	j	N
0	0	5
1	1	3
2	2	1
3	3	-1

pseudo-code.

```
void  spiral Printing ( arr, N) {
        i = 0,  j = 0

        while ( N > 1 ) {
            //1. print (N-1) elements from l → r
                for ( k = 1 ;  k < N ;  k++) {
                        print( arr[i][j] ; -
                        j ++
                }

            //2. print (N-1) elements from t to d.
                for ( k = 1 ;  k < N ;  k++) {
                        print( arr[i][j] ;
                        i ++
                }

            //3. print (N-1) elements from r to l
                for ( k = 1 ;  k < N ;  k++) {
                        print( arr[i][j] ;
                        j --
                }

            //4. print (N-1) elements from d to t
                for ( k = 1 ;  k < N ;  k++) {
                        print( arr[i][j] ; -
                        i --
                }

            i ++,  j ++,  N -= 2 ;
        }

        if (N == 1){  print (arr[i][j])}
}
```

T.C → $O(N^2)$

S.C → $O(1)$

Doubt.

[min no. of switch pressed ?]

bulbs → 1 1 0 1 0 0 1 0 1

1 1 1 0 1 1 0 1 0

1 1 1 1 0 0 1 0 1

1 1 1 1 1 1 0 1 0

1 1 1 1 1 1 1 0 1

1 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1

(observation.)

1 1 1

1 1 ~~0~~ ~~1~~ ~~0~~ 0 1 0 1

[0] [0]

0 1 2 3 4 5 6 7 8

Count = ~~0~~ ~~1~~ ~~2~~ 3

```
count=0;
for( i=0 ; i< n; i++) {
        if ( count % 2 ==0) { state = arr[i] }
        else               { state = 1- arr[i] }

        if (state ==0) {
                count++;
        }
}
```

0 → 1
1 → 0
?

arr→	1	1	0	1	0	0	1	0	1
	0	1	2	3	4	5	6	7	8

arr→	✓	✓	x	x	✓	x	✓	x
	C	a	d	B	E	m	I	K
	0	1	2	3	4	5	6	7

N = 8.

N.

total no. of subarrays starting with idx → 0 ⇒ (8) N

" " " " " " idx → 1 ⇒ (7) N-1

" " " " " " idx → 2 ⇒ (6) N-2

|

|

idx → i ⇒ N-i

ans = (7) + (4) + (2) = 13.