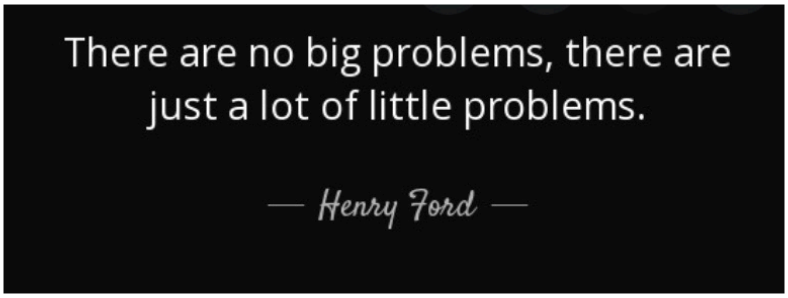
Today's Quote →



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.

$\boxed{K=4}$

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

$\boxed{N=9}$

total Sub.arrays of length K?

	K=1.	K=2	K=3	----	K.
total no. of subarrays of size-K	N	N-1	N-2		$\boxed{N-K+1}$
	[9]	[8]	[7]		

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

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

arr	-3	4	-2	5	3	-2	8	2	-1	4
index	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

ans : 16

idea1: For every sub-array of len K, iterate and get the sum. Overall max. sum will be ans.

```
int maxSubarray ( arr, N, K) {
    s = 0 , e = K-1 , ans =
    while( e < n ){
        sum = 0        //calculating sum [s,e]
        for( i -> s to e) {
            sum += arr[i]
        }
        if (sum > ans)  ans = sum
        s += 1 , e += 1 ;
    }
    return ans ;
}
```

$T.C \rightarrow (n-k+1) * K \Rightarrow (n-\frac{n}{2}+1)(\frac{n}{2}) \Rightarrow (\frac{n}{2}+1)(\frac{n}{2})$

$= \frac{n^2}{4} + \frac{n}{2} \rightarrow O(n^2)$

$K=1$ $\quad$ $K=n$ $\quad$ $K=n/2$

$TC \rightarrow O(n)$ $\quad$ $O(n)$

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

idea-2.

Use prefix.sum to optimise the inner loop.

step-1 Create pSum[ ]

step-2. s = 0 , e = K-1 , ans = MIN-VALUE

(// Google what is the min value in your long.)

```
while( e < n ){
    //calculate sum of subarray [s, e] .
    sum = 0
    if ( s == 0)   sum = p Sum[e]
    else           sum = psum[e] - pSum [s-1]

    if (sum > ans)  ans = sum

    s += 1 , e+=1 ;
}
return anw;
```

$\boxed{T.C \to O(n) \quad , \; S.C \to O(n)}$

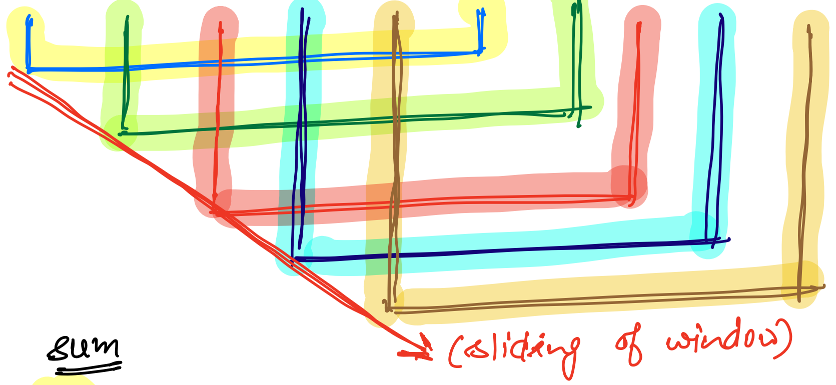
idea-3 :

$\boxed{K=6}$

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

0 1 2 3 4 5 6 7 8 9

$\boxed{N=10}$



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

carry forward + $\overset{\text{All}}{\text{subarrays}}$ of same size ⇒ $\boxed{\text{Sliding Window}}$

Final code :

```
int  maxSum( arr, N , K) {
    // calculate sum of 1st K elements
    sum = 0
    for( i → 0  to  K-1) {
            sum += arr[i]
    }                                   K iterations
    ans = sum
    s = 1 , e = K

    while ( e < N ) {
        //get subarray sum from [s, e].
        sum = sum - arr[ s-1] + arr[e]            N-K
        if ( sum > ans)   ans = sum               iterations

        s += 1
        e += 1
    }
    return  ans
}
```

T.C → O(N)
S.C → O(1)

Break till 10:30

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 $\boxed{5}$ } , $\boxed{B = 8}$

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

$\boxed{ans = 2}$

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

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

$\boxed{ans = 1}$

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

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

$\boxed{ans = 1}$

→ count of all element < B [count]

→ size of sub-array is fixed ⇒ count

→ We need to find a sub-array in which swaps are min.

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

All elements greater than B → bad elements.
〃 〃 smaller than B → good elements.

<u>pseudo-code</u>.

```
int minimumSwaps( arr, N, B) {
    //1. find count of good elements.
    count = 0
    for( i → 0 to N-1) {
        if (arr[i] < B) count += 1
    }
    if (count == 0 || count == 1) return 0

    //2. find bad elements in first subarray of
          size = count.
    bad = 0
    for( i → 0 to count-1) {
        if (arr[i] ≥ B) bad += 1
    }
    //3. Apply sliding window technique
    ans = bad , s = 1 , e = count
    while ( e < N) {
        if ( arr[s-1] ≥ B) bad -= 1      (removing arr[s-1])
        if (arr[e] ≥ B ) bad += 1        (adding arr[e])
        if ( bad < ans) ans = bad
        s += 1 , e += 1
    }
    return ans;
}
```

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	2 ✓	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 ↑

```
void printBoundary ( mat , N ) {
    i = 0  , j = 0
    //1. print N-1 elements from l tor
        for ( K ⟶ 1 to N-1) {
            print (arr[i][j])
            j += 1
        }
    //2. print N-1 elements from t to d
        for ( K → 1 to N-1) {
            print (arr[i][j]
            i += 1
        }
    //3. print N-1 elements from r to l
        for (K → 1 to N-1) {
            print( arr[i][j])
            j -= 1
        }
    //4. print N-1 elements from d to t
        for (K → 1 to N-1) {
            print (arr[i] [j])
            i -= 1
        }
}
```

K	i	j
1	0	0
2	0	1
3	0	2
4	0	3
	0	4

4, 4

4, 0

0, 0.

T.C → O(N)
S.C → O(1)

	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	-2
1	1	4
		-2
2	2	2
3	3	0

Spiral Printing.

	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
		-2
1	1	3
2	2	1

N-1 = 1-1=0

```
void   printBoundary ( mat , N ) {
        i = 0   , j = 0
      while ( N > 1  ) {
          for ( K → 1   to  N-1) {
                  print (arr[i][j])
                   j += 1
          }

          for ( K → 1   to  N-1) {
                  print (arr[i][j]
                   i += 1
          }

          for (K → 1   to  N-1) {
                  print (arr[i][j])
                  j -= 1
          }

          for (K → 1  to  N-1) {
                  print (arr[i][j])
                   i -= 1
          }
            i  += 1  ,  j += 1 ,   N -= 2
      }
        if (N==1)   print arr[i][j]
      }
```

T.C → $O(N^2)$

S.C → $O(1)$

Doubts →

→ {Complete all your H.W/ Assignments questions. }

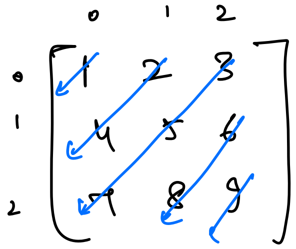
24. hrs

arr →

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 2 & 3 & 0 & \boxed{0} & 15 \\
1 & \boxed{0} & 5 & 19 & 16 & 22 \\
2 & 6 & 20 & 9 & 14 & 7 \\
3 & 11 & \boxed{0} & 10 & 8 & 21 \\
4 & 18 & 12 & 17 & 23 & 13
\end{array}$$

$$a1 \rightarrow \begin{array}{|c|c|c|c|c|}
\overset{t}{\cancel{f}} & \overset{t}{\cancel{f}} & f & \overset{t}{\cancel{f}} & f \\
0 & 1 & 2 & 3 & 4
\end{array} \text{(rows)}$$

$$a2 \rightarrow \begin{array}{|c|c|c|c|c|}
\overset{t}{\cancel{f}} & \overset{t}{\cancel{f}} & \overset{t}{\cancel{f}} & \overset{t}{\cancel{f}} & f \\
0 & 1 & 2 & 3 & 4
\end{array} \text{(cols)}$$



$$\begin{bmatrix}
1 & 0 & 0 \\
2 & 4 & 0 \\
3 & 5 & 7 \\
6 & 8 & 0 \\
9 & 0 & 0
\end{bmatrix}$$

```
boolean isPrime( int A){
        for(int i=2 ; i*i <=A ; i++){
                if(A % i ==0) return false
        }
        return true;
}
```

√A iterations

```
int solve (A){
        int count =0
        if (A >=2) count++;
        for(i = 3 ; i <=A ; i+=2){
                if(isPrime(i)) count++
        }
        return count;
}
```

3, 5, 7, 9, — — - A. ⇒ $A/2$ iterations.

3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, / —— A] ⇒ A iterations,

$$\begin{array}{l|c} & \text{iteration.} \\ \hline i = 3 & \sqrt{3} \\ & + \\ i = 5 & \sqrt{5} \\ & + \\ i = 7 & \sqrt{7} \\ & + \\ i = 9 & \sqrt{9} \\ & + \\ | & | \\ | & | \\ & + \\ i = A & \sqrt{A}. \\ \hline \end{array}$$

total no. of iteration : $\underbrace{\sqrt{3} + \sqrt{5} + \sqrt{7} + \text{———} \sqrt{A}}_{A/2}$.

$$\Rightarrow \boxed{\frac{A}{2} \cdot \sqrt{A}} = \underline{O(A\sqrt{A})}.$$