Arrays : Sliding window

Jul 10, 2023

AGENDA

- Sliding window concept
- 2 problems on sliding window
- 1 problem on 2D Matrices

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

N = 10

K = 5

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

0 K-1

= 7

ans = -∞

ans = max (ans, curr)

start		end	sum
0	-	K-1	7
1	-	K	8
2	-	6	12
3	-	7	16
4	-	8	10
5	-	9	11

Ans = 16

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

Brute Force

* Get all subarrays of length 'K'.

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

s, e

i

s = 0, e = 4

```
s = 0
e = k-1
ans = INT_MIN
while (e < n)            <- n-k+1
{
        // find sum of s,e.
        sum = 0                                  <- O(K)
        for(int i=s; k<=e; i++)
              sum += arr[i]

        ans = max(ans, sum)      if (sum > ans)
                                     ans = sum

        s++; e++;      <- go to the next window.

}
print(sum);
```

T.C. -> $\boxed{O(K * (N-K+1))}$

$= O(N^2)$ ↓

S.C. -> O(1)

See below.
(No. of subarrays)

* How many subarrays of len K ?

$N=10$
$K=5$

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

$10-5+1$

$N-K+1$

1st window : st = 0 , end = K-1

2nd window : st = 1 , end = K

3rd window : st = 2 , end = K+1

4th window : st = 3 , end = K+2

⋮

last window = st = n-K , end = n-1

a-b+1

$(n-1)-(n-K)+1 = K$

{0, 1, 2, 3 . . . , n-k} → How many elements?

$\boxed{n-K+1}$

n-K+1 subarrays on lenk are present.

$$O(K * (N-K+1))$$

window size:

$K = 1$

$$1 * (N-1+1) = O(N)$$

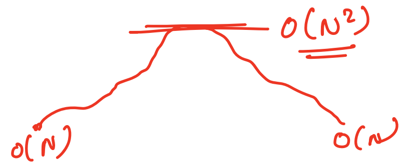
$K = N/2$

$$\frac{N}{2} * \left(N - \frac{N}{2} + 1\right) = \frac{N^2}{4} = O(N^2)$$

$K = N$

$$N * (N-N+1) = O(N)$$

$K = 1 \ldots\ldots N/2 \ldots\ldots N$



Prefix sum

↳ Get rid of inner loop.

```
// Create a pf array.

s = 0
e = k-1

ans = INT-MIN
while ( e < n )          ← O(N-K+1)
{
    // find sum of s,e.
    if(s != 0)
        sum = pf[e] - pf[s-1]
    else
        sum = pf[e]

    ans = max(ans, sum)

    s++; e++;            ← go to the next window.

}
print(sum);
```

T.C. → O(N-K + N)

= O(2N-K)

= O(N)

S.C. = O(N)

prefix array.

Not happy :(

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

s	e	sum	
0	4	$-3 + 4 + (-2) + 5 + 3$	$= 7$
1	5	$4 + (-2) + 5 + 3 - 2$	$= 7 - (-3) + (-2)$ $= 8$
2	6	$-2 + 5 + 3 +$ outgoing = 4 incoming = 8	$= 8 - 4 + 8$ $= 12$
3	7	outgoing = -2 incoming = 2	$12 - (-2) + 2$ $= 16$

$8 \quad -3 + 7 = 12$

$19 - 0 + 1 = 20$

3 1 0 4 7 8 1 -1 2 6

$K = 4$

8

$12 - 1 + 8$

$= 19$

Sliding window : Window size is fixed and window can slide to the right.

Make use of the result of the previous window.

Code.

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

// sum of elements in 1st window

K ←

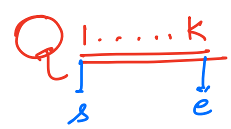
```
sum=0
for (int i=0; i<k; i++)
        sum += arr[i]
```

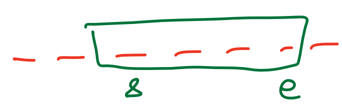
```
ans=sum
```

// slide window to the right.

n-K ←

```
s = 1 , e = k          ; start and end of 2nd window.
while( e < n)
{
        out = arr[s-1]
        in = arr[e]
        sum = sum - out + in
        ans = max (ans, sum)
        s++
        e++
}
```





T.C. → O(K + n - K)

= O(N)

S.C. = O(1)

window size is fixed.

Q. Given N array elements, find min no. of swaps reqd. to bring all elements <= B together.

1, 12, 10, 3, 14, 10, 5 B = 8

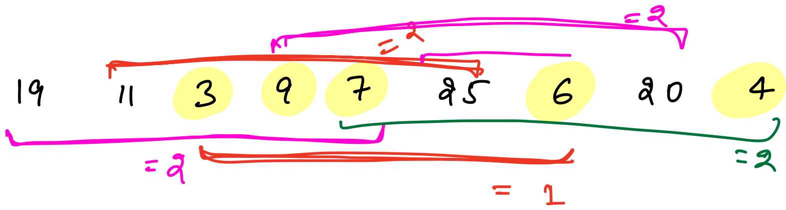
ans = 2

19 11 3 9 7 25 6 20 4 B = 10

ans = 1

25 30 2 18 7 6 9 50 3

ans = 1



Approach

Good guys → <= B
Bad → > B

* Calculate no. of good guys. = K
* Consider sliding window size 'K'.
* Find no. of bad guys in each window.

= no. of swaps reqd. for this window to host all good guys.

code.

```
good=0
for(int i=0; i<n; i++)
{
        if( arr[i] <= B)
                good++;
}

K = good        ; // window size
```

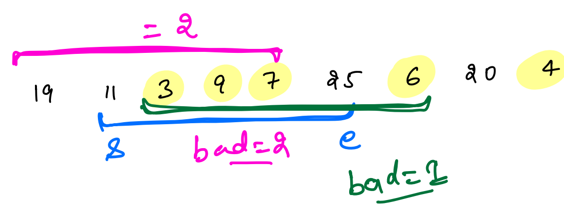
// Find no. of bad in 1st window.

```
bad = 0
for(int i = 0; i < K ; i++)
{       if (arr[i] > B)
            bad++;

}
```

// No. of bad elements = no. of swaps reqd for this window.

```
ans = bad;
```

// Slide window.



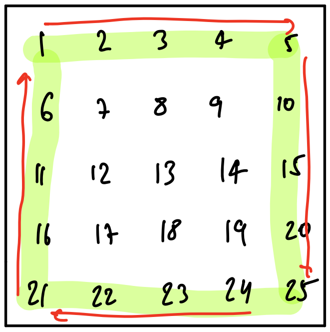

```
s = 1 , e = K ;
while (e < n)
{
        out = arr[s-1]
        in = arr[e]

        if (in > B)
              bad++ ;
        if (out > B)
              bad-- ;

        ans = min(ans, bad) ;

        s++
        e++

}
return ans;
```

ans = 10
bad = 5

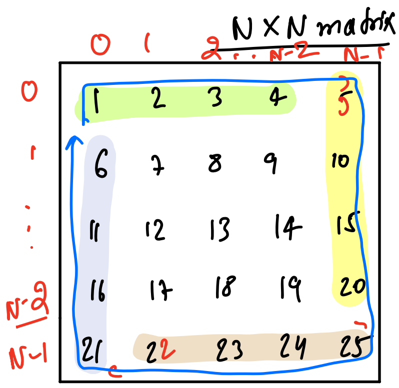
T.C. → O(N)
S.C. → O(1)

Break till 8:40 AM

Q: Given a matrix of N×N, print the boundary of matrix in clockwise direction.



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



1st leg → 0,0 → 0, N−2

2nd leg. 0, N−1 → N−2, N−1

3rd leg N−1, N−1 → N−1, 1

4th leg. N−1, 0 → 1, 0

* Same no. of elements in each leg.

No. of elements in each leg = N−1

<u>Code</u>

i=0, j=0

// Print 1st leg.

```
for(int K=1 ; K<=N-1 ; K++)   // Run loop N-1 times
{
        print (arr[i][j])
        j++
}
```

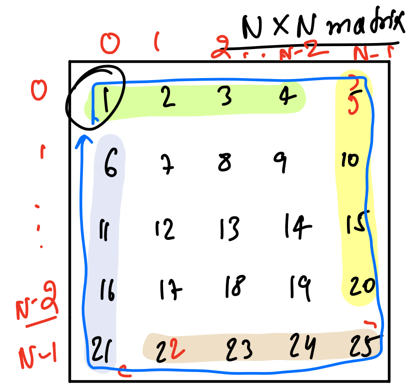
// Value of i is 0, j is N-1.

// Print 2nd leg.

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

// Value of i is N-1, j is N-1.

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

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



// value(i,j) becomes 0,0.

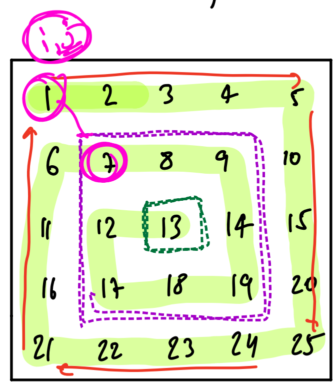
T.C. → O(N)

Q: Print the matrix in a spiral order.

N*N

↓

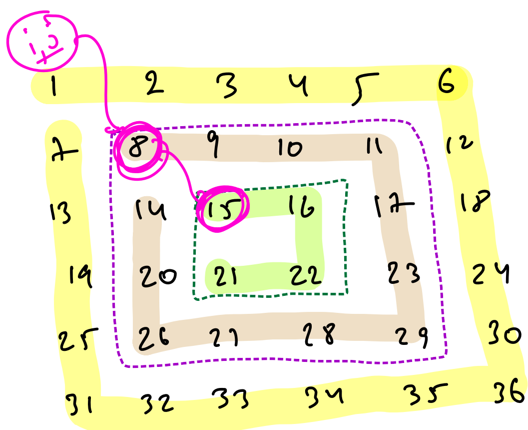
N-2*(N-2)



6×6

↓

4×4



Code

i=0, j=0

```
while ( N > 1 )
{
    // Print 1st leg.
    for(int K=1 ; K<=N-1; K++)
    {
        print (arr[i][j])
        j++
    }
    // Value of i is 0, j is N-1.
    // Print 2nd leg.
    for(int K=1 ; K<=N-1; K++)
    {
        print (arr[i][j])
        i++;
    }
    // Value of i is N-1, j is N-1.
    for(int K=1 ; K<=N-1; K++)
    {
        print (arr[i][j])
        j--;
    }
    for(int K=1 ; K<=N-1; K++)
    {
        print (arr[i][j])
        i--;
    }
    i++
    j++
    N = N-2
}
if (N== 1)
    print ( arr[i][j])
```

: Run loop N-1 times.

5x5 → 3x3 → 1x1 → $\boxed{1}$

2x2 matrices and greater

$\boxed{6\times6 \to 4\times4 \to 2\times2}$ → 0x0

or print (arr [N/2] [N/2])

T.C. ⇒ $O(N^2)$