Sliding Window

Vinay Neekhra

Senior Instructor & Mentor

Reachable in Scaler Lounge

"Let’s see what’s out there" - Dr. Ravi Kothari

Content: ① Sliding window
② 2 problems

Quiz N elements, indices for last subarray of len=K

① K, N-1 → 1, 0 ✗
② N-K, N-1 → 0, 0
③ N-K+1, N-1, 1, 0 ✗
④ None, None

N = 1
K = 1

[9] → (0,0)

5. A = [4 | 5 | 6 | 9 | 11 | 13 | -1 | 2]
0 1 2 3 4 5 6 7

first subarray size is 3 (0,2) index

second subarray → (1,3) → [5 | 6 | 9]

3rd ——— [6 | 9 | 11]

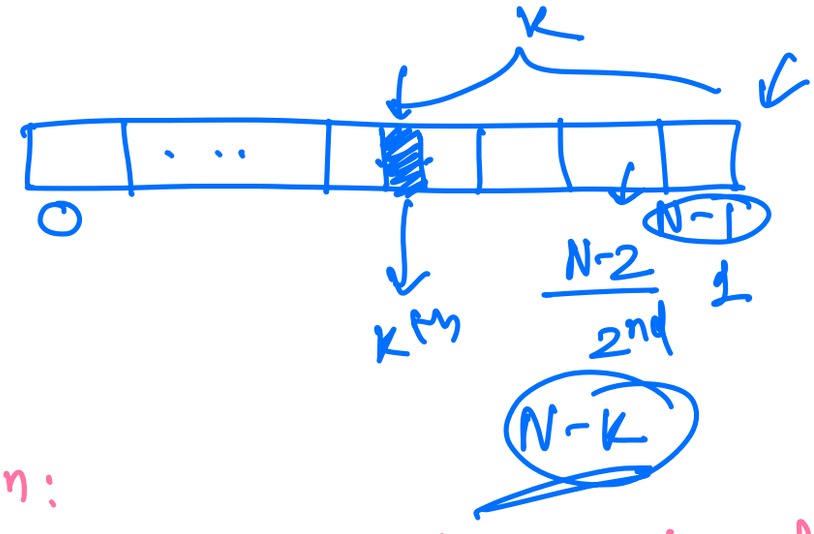
A = [10 | 20 | 90]

K=1 ⇒ {[10], [20], [90]}

N	K	#Subarrays
3	1	3

K=2 ⇒ { [10|20], [20|90] } 3 2 2

K=3 ⇒ { [10|20|90] } 3 3 1



N-K+1

[L, R] ⇒ R-L+1

[2, 9] ⇒ 9-2+1 ⇒ 8

Obsⁿ:

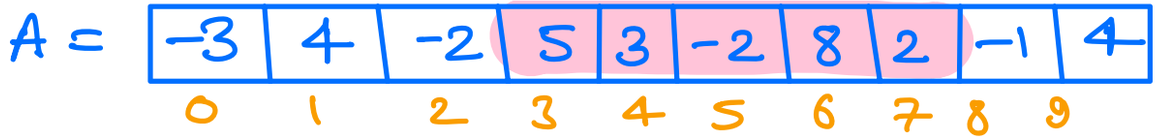
Starting index varies from [0 → N-K]

How many numbers

N-K+1

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

ex:



ans = 16

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

ans = 16

(I) Brute force : Explore all the possibilities.

idea: for every subarray of size K iterate & calculate the sum and find the max.

```
int maxSubarraySum (A, K){
    S=0, e=K-1, ans = INT_MIN
    while (e<N) {                          N-K+1
        int sum=0;
        for (i=S; i<=e; i++){              K
            sum += A[i]
        }
        if (sum>ans){
            ans = sum
        }
```

T.C. = (N-K+1) * K

K=1 : N*1, T.C. = O(N)

K=N : 1*N, T.C. = O(N)

$k = N/2$

$(N - \frac{N}{2} + 1) * (\frac{N}{2})$

$\Rightarrow (\frac{N}{2} + 1) * (\frac{N}{2})$

$\Rightarrow O(N^2)$

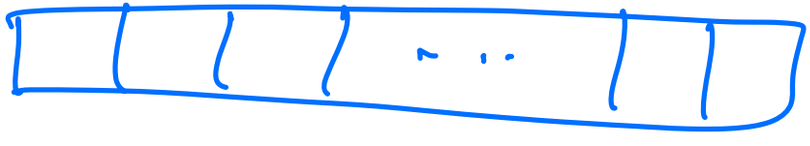
T.C. = $O(N^2)$

S.C. = $O(1)$

```
        s++;
        e++;
    }
    return ans
}
```

II nd Soln : Prefix Sum:

// prefix sum array pSum[N] → O(N) time

pSum = 

```
int maxSubarraySum (A, K) {
    S = 0, e = K-1, ans = INT_MIN
    while (e < N) {
        int sum;
        if (S > 0) {
            sum = pSum[e] - pSum[S-1]
        }
        if (S == 0) {
            sum = pSum[e]
        }
        if (sum > ans)
            ans = sum
        S++;
        e++;
    }
    return ans
}
```

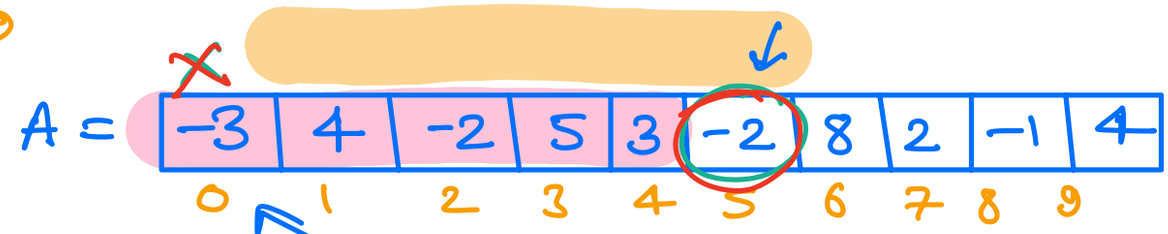
N

N-K+1 time

T.C. = O(N)

S.C. = O(N)

III →



S e sum

0 4 7

1 5 10-2 = 8

2 6 12

Sum = A[0] + A[1] + A[2] + A[3] + A[4]

Sum = sum − A[0] + A[5]

Sum = sum − A[1] + A[6]

3 7 Sum = sum − A[2] + A[7]

S, e Sum = sum − A[s-1] + A[e] → s=0? → sum → (0 → k-1)

[carry forward + All subarrays of size K] ⇒ sliding window

⇒ Partial sum is carrying forward

Final code:

obvⁿ K>N ✗

```
int maxSubarraySum (A, K) {

    //1. Calculate the sum of first K elements.
       sum = 0
       for (i=0; i<K; i++){
            sum += A[i]
       }                                  ⇒ K times

       ans = sum, s = 1 , e = K

       while (e<N){                       ⟹ N-K+1
            // Calculate the sum of this subarray

            sum = sum - A[s-1] + A[e]
            if (sum > ans)
                 ans = sum
            s++
            e++
       }
}
```

T.C. = O(K + N-K+1)
= O(N)

return ans

}

S.C. = O(1)

Most optimal complexity

Q. Given an array of N elements, find and return minimum no. of swaps to bring all the numbers <= B together.

ex. A = | 1 | 12 | 10 | 3 | 14 | 10 | 5 |

B = 8

ans = 2

10:35

Quiz: A = | 19, | 11, | 3, | 9, | 7, | 25, | 6 | 20 | 4 |

B = 10

19, 11 3, 9, 7, 4, 6 20 25

Quiz: A = [25, 30, 2, 18, 7, 6, 9, 50, 3]

B = 10

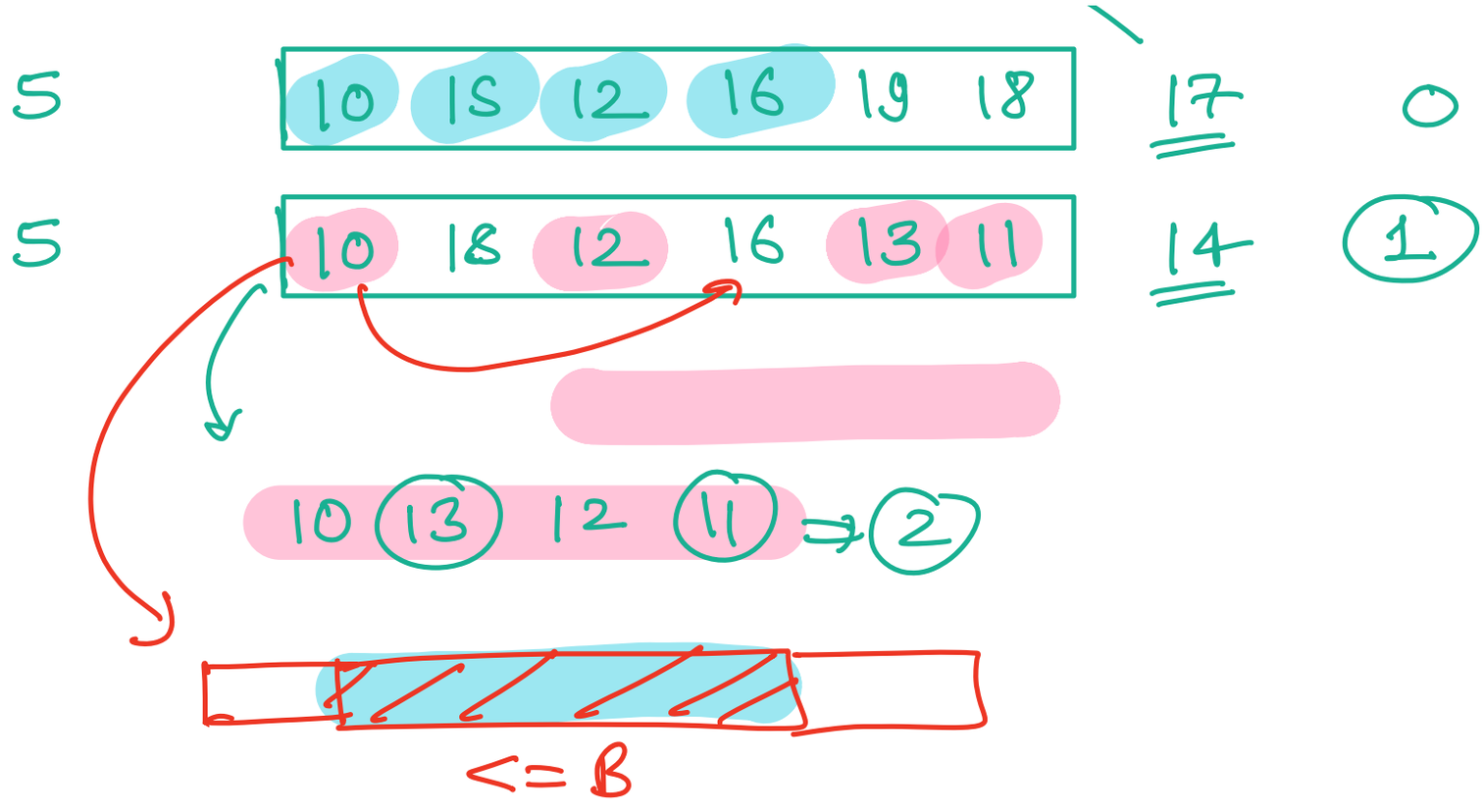
I [25, 30, 2, 3, 7, 6, 9, 50, 18]

ans = 1

(II) [25, 30, 50, 18 7, 6, 9, 2, 3] ans = 1

(I) obsn

N	array	B	ans
0	[]	10	0
0	[]	X	0
1	[10]	5	0
1	[10]	15	0
1	[9]	X	0
2	[10,20]	5	0
2	[10, 20]	15	0
2	[10,20)	30	0
3	[10,20,30]	12	0
3	10, 20, 30	22	0
3	[30, 10, 20]	15	0
3	[10 30 20]	25	1



Obs^{n}:

(I) the group size which would contain all the elements <= B, would be equal to the number of elements <= B (K)

(II) find the subarray for which min swaps are needed

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

K = 5 (elements <= 10)

III Swaps == bad elements in the subarray
(> B)

IV Good elements <= B in the subarray

```
int minSwaps (A, B) {
    // find #elements <= B

    K = 0
    for (i=0; i<N; i++) {
        if (A[i] <= B) {
            K++;
        }
    }
```

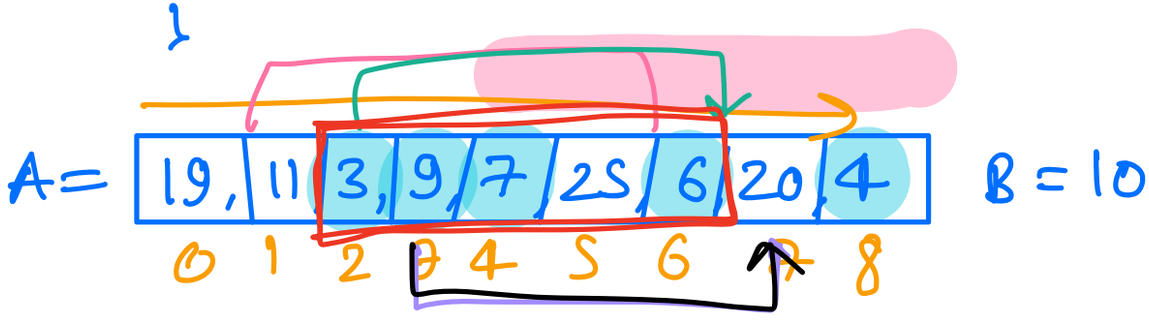
good elements

```
}
// k denotes how many numbers <= B
// find good element in the first subarray of size k
good = 0
for (i = 0; i < k; i++) {
    if (A[i] <= B) good++;
}
ans = k - good
s = 1, e = k
while (e < N) {
    if (A[e] <= B) good++;
    if (A[s-1] <= B) good--;

    swap = k - good        // bad element which need to be swapped

    if (swap < ans) {
        ans = swap
    }
    s++
    e++
```

T.C. = O(N)

S.C. = O(1)

```
    }
    return ans;
}
```



K = 5

good = 3 → 4 3 . . . 3

swap = 2 1 2 . . . 2

ans = 2 1 .

Doubt session

(I) Office hours : (1) 8th June 9 to 11 pm
→ Have atleast 1 question

(II) Contest : (1) Revise till last week

(2) Time complexity & array techniques
- Prefix sum
- contribution
- carry forward

(3) ☒ sliding window won't be part of this contest!

(4) Friday - contest — 9th June

(5) Programming

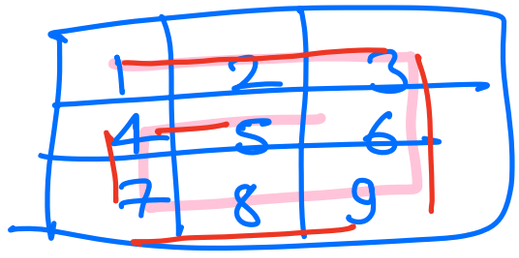
→ The more sweat during the preparation, the less blood during the war!

90 minutes (~5)

(6) Contest reattempt — (I) give (II) reattempt

(7) easy, med, ~~Hard~~

Q. Spiral printing: NXN

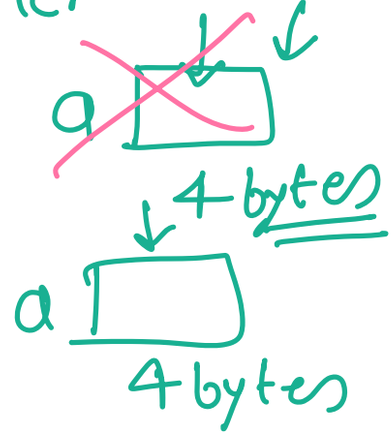


⇒ ans 1, 2, 3, 6, 9, 8, 7, 4, 5

Pramp.com → mock interviews → 90%

```
while ( ) {
    int a = 0
    :
}
```

compiler



attendence: not getting updated → (after the Monday outage)

PSP : ~