Array : Sliding Window

Problem Solving Session (Tomorrow 9-11 PM)

- ↳ Problems given in assignments/ homeworks which are least solved by students
- ↳ idea & pseudocode will be discussed
- ↳ optional , attendence not counted , 2- 2.5 hrs
- ↳ recorded

Question 1

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

eg

$$A[10] = \begin{array}{cccccccccc} -3 & 4 & \boxed{-2 \quad 5 \quad 3 \quad -2 \quad 8} & & & & & 2 & -1 & 4 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

$K=5$

for first subarray of range [s, e]

$S=0$

$e-s+1 = K \Rightarrow e = K-1$

for last subarray of range [s, e]

$e=n-1$

$e-S+1 = K \Rightarrow n-\cancel{1}-S\cancel{+1} = K \Rightarrow s = n-K$

Code

```
def subarraySum ( a[] , K ) {
        n = a.length                    write in your language
                                            ↓
        s=0, e=K-1 , ans= INT_MIN ~~ans=0,~~ / ~~a[0]~~ / Σ ✓ / ~~-1~~
n-K+1
iterations  → while( e < n ) { ⟺ [S <= n-K]
              sum=0
K iterations →  for ( i=s; i<=e ; ++i ) {
                    sum += a[i]
                }
                if( sum > ans )
                    ans = sum
                s++ , e++
        }
        return ans
}
```

TC : O(K*(n-K+1))

SC : O(1)

start index of first subarray = 0

start index of last subarray = n-K

no. of subarrays = n-K -0 +1

= n-K+1

[1 . . . . . N]

N -1 +1

= N

TC: $O(K\cdot(n-K+1))$

if K=1

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

$O(N)$

if K=N

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

$O(N)$

if K=N/2

$O(n/2\,(n-n/2+1))$

$O(n/2\,(n/2+1))$

$O(n^2/4 + n/2)$

$O(N^2)$

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

$O(nK - \cancel{K^2} + \cancel{K})$

$O(nK)$

TC: $O(N^2)$

SC: $O(1)$

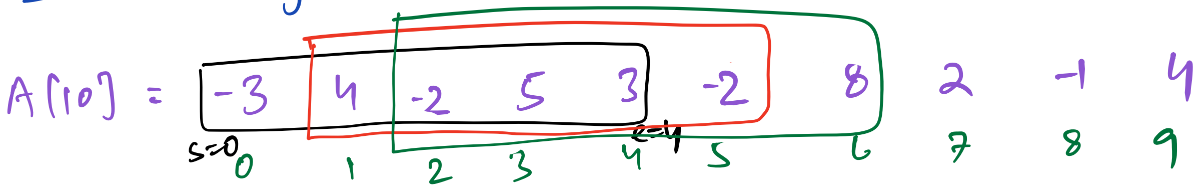
Idea 1 : Prefix Sum

TC: $O(N+N) = O(N)$

SC: $O(N)$

→ TODO

Idea 2 : Carry forward aka Sliding Window



K=5

$sum_1 = 7$ s=0, e=4

$sum_2 = 7 - (-3) + (-2) = 8$ s=1, e=5

a[s-1] a[e]

$sum_3 = 8 \underset{a[s-1]}{- 4} \underset{a[e]}{+ 8} = 12$ $s=2, e=6$

⋮

Code

```
int subarraySum (a[], k) {
    n = a.length
    sum=0
    for(i=0 ; i<K; ++i) {      → subarray [0, k-1]
        sum += a[i]
    }
    ans = sum
    s=1 , e=K
    while ( e<n) {
        // get subarray sum from [s, e]
        sum = sum - a[s-1] + a[e]
        if (sum > ans)
            ans = sum

        s++, e++
    }
    return ans
}
```

K iterations → (for loop)

n-K → iteration (while loop)

O(n-K+K)

TC : O(N)

SC : O(1)

Question 2

Given an array A and iteger B, find minimum swaps required to bring all numbers <= B together.

eg A = 1 12 10 3 14 10 5

B=8

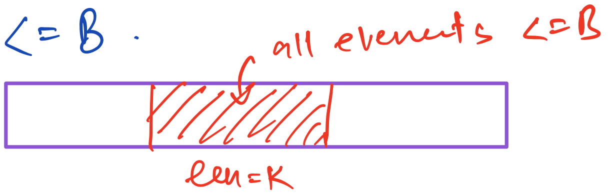
10 12 1 3 5 10 14

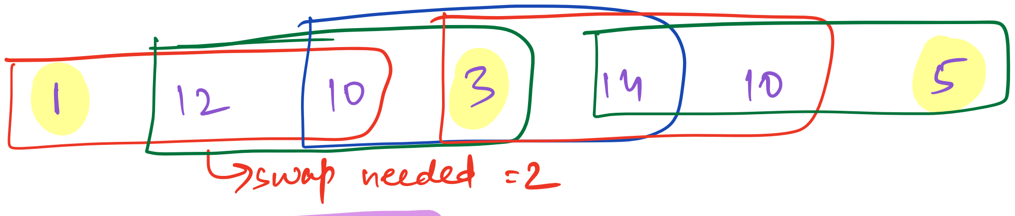
#swaps = 2

5 17 100 11 B = 20

5 17 11 100 #swaps=1

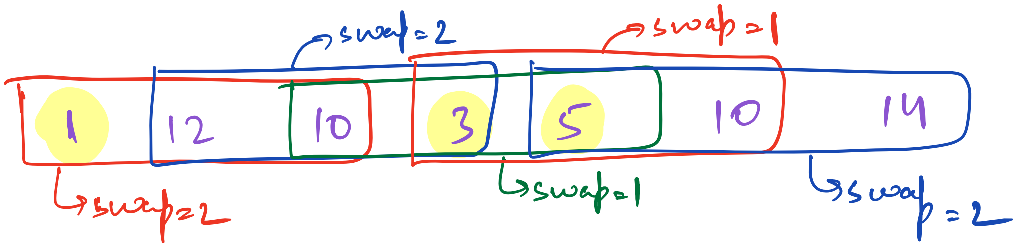
Idea : let say there are K elements which are <= B.





B=5   K=3

min. swap needed =2



min. swap needed =1

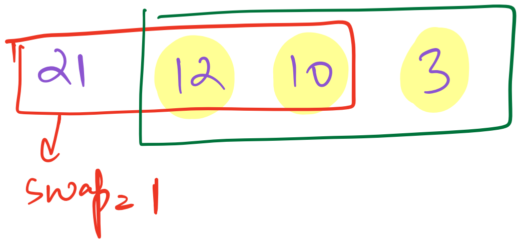
Code

```
int  minSwaps( a[], B) {
    n = a.length
    K=0  ⟶ size of subarray
    for(i=0; i<n; ++i) {
        if (a[i] <= B)
            ++K
    }

    s=0,  e=K-1 ,  ans=K

    while( e<n) {
        swap=0
        for(i=s; i<=e; ++i ){
            if (a[i] > B)
                ++swap
        }
        if(swap < ans)    ans=swap
        ++s , ++e
```

TC: $O(N^2)$

SC: $O(1)$

```
    }
    return ans
}
```



B = 15

$a[s-1]$	$a[e]$	
$<= B$	$<= B$	$swap_2 = swap_1$
$<= B$	$> B$	$swap_2 = swap_1 + 1$
$> B$	$<= B$	$swap_2 = swap_1 - 1$
$> B$	$> B$	$swap_2 = swap_1$

$$swap = swap - f(a[s-1]) + f(a[e])$$

```
f(x) :   if (x > B) ret 1
         else  ret 0
```

$a[s-1]$	$a[e]$	$f(a[s-1])$	$f(a[e])$	
$<= B$	$<= B$	0	0	swap = swap
$<= B$	$> B$	0	1	= swap + 1
$> B$	$<= B$	1	0	= swap - 1
$> B$	$> B$	1	1	= swap

Code

```
int minSwaps (a[], B) {
    n= a.length
    K=0
    for( i=0; i<n; ++i ) {
        if( a[i] <=B)
            ++K
    }
    swap =0
    for( i=0; i<K; ++i) {
        if (a[i] >B)   ++swap
    }
    ans = swap
    s=1 , e =K
    while ( e<n) {
      // find swap in subarray [s, e]
      if( a[s-1] >B)
            --swap
      if (a[e] >B)
            ++swap
      if (ans > swap)   ans = swap
      s++, e++
    }
} return ans
```

TC : O(N)

SC : O(1)