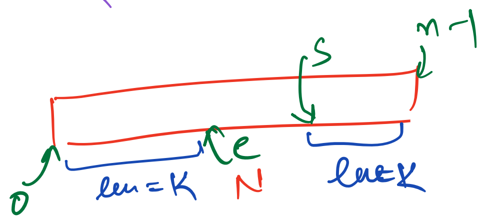
Array: Sliding Window

Question 1

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

A[10] = -3 4 -2 5 3 -2 8 2 -1 4

K = 5



first subarray = [0, K-1]

last subarray = [n-K, n-1]

$e - 0 + 1 = K \Rightarrow e = K-1$

$(n-\cancel{1}) - s + \cancel{1} = K \Rightarrow s = n-K$

```
int subarraySum( int a[], int K) {
    int n = a.length                          ↙ write it in your
                                                own language
    int s=0, e = K-1;      ans = INT_MIN
    while (e<n ) { or s <=n-K
        sum=0;
        for (i=s;  i<=e; ++i ) {
            sum = sum + a[i];                 } → K iterations
        }
        if (sum > ans)   ans = sum;
        ++s, ++e;
    }
    return ans;
}
```

Start index of first subarray = 0

start index of last subarray = n-K

[1 ... N]

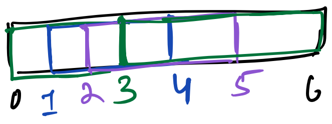
$\Downarrow$

N-1 +1

N

# of subarrays = n-K -0+1

= n-K+1



N=7 # subarrays = 4

K=4

$1 \le K <= N$

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

$\Rightarrow$ ~~$n \times K - K^2 + K$~~

$\Rightarrow$ ~~$n \times K$~~

K=1

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

$O(n)$

K=N

$O(N(\cancel{N}-\cancel{N}+1))$

$O(n)$

K=N/2

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

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

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

$O(n^2)$

TC: $O(n^2)$

SC: $O(1)$

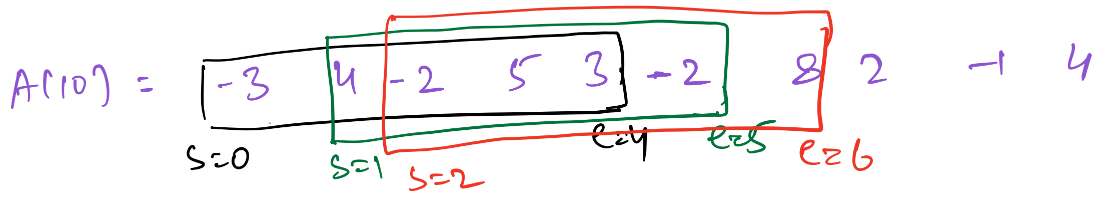
Idea 1: Prefix Sum $\Rightarrow$ TODO

TC: $O(N+N)$ $\Rightarrow$ $O(N)$

SC: $O(N)$

Idea 2 : Carry forward aka Sliding Window

$K=5$



$s=0 , e=4$

$sum_1 = 7$

$sum_2 = 7 - \underset{a[s-1]}{(-3)} + \underset{a[e]}{(-2)} = 8$ $\quad s=1 , e=5$

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

⋮

```
int subarraySum( int a[] , int K ) {
    int n = a.length        sum=0
    for ( i=0; i<K; ++i ) {        → subarray [0,K-1]
        sum =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;
}
```

TC: O(N)

SC: O(1)

Question 2

Given an array A and integer 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

eg 5 17 100 11 B = 20

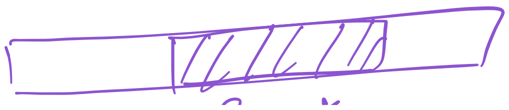
5 17 11 100 \# swaps = 1

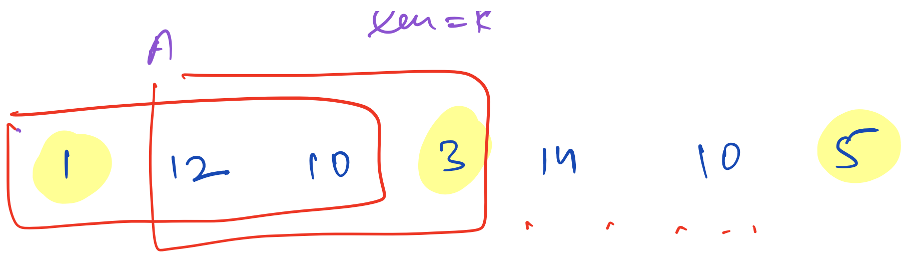
5 17 100 11

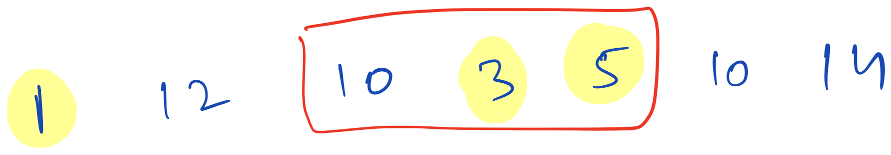
100 17 5 11 \# swaps = 1

Idea

there are K elements which are <= B.





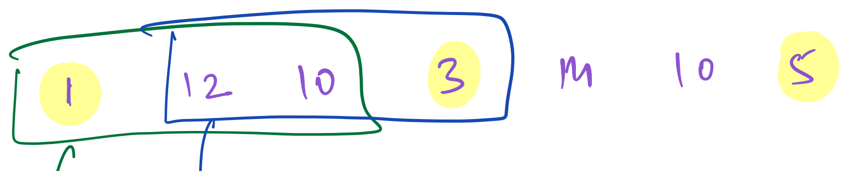


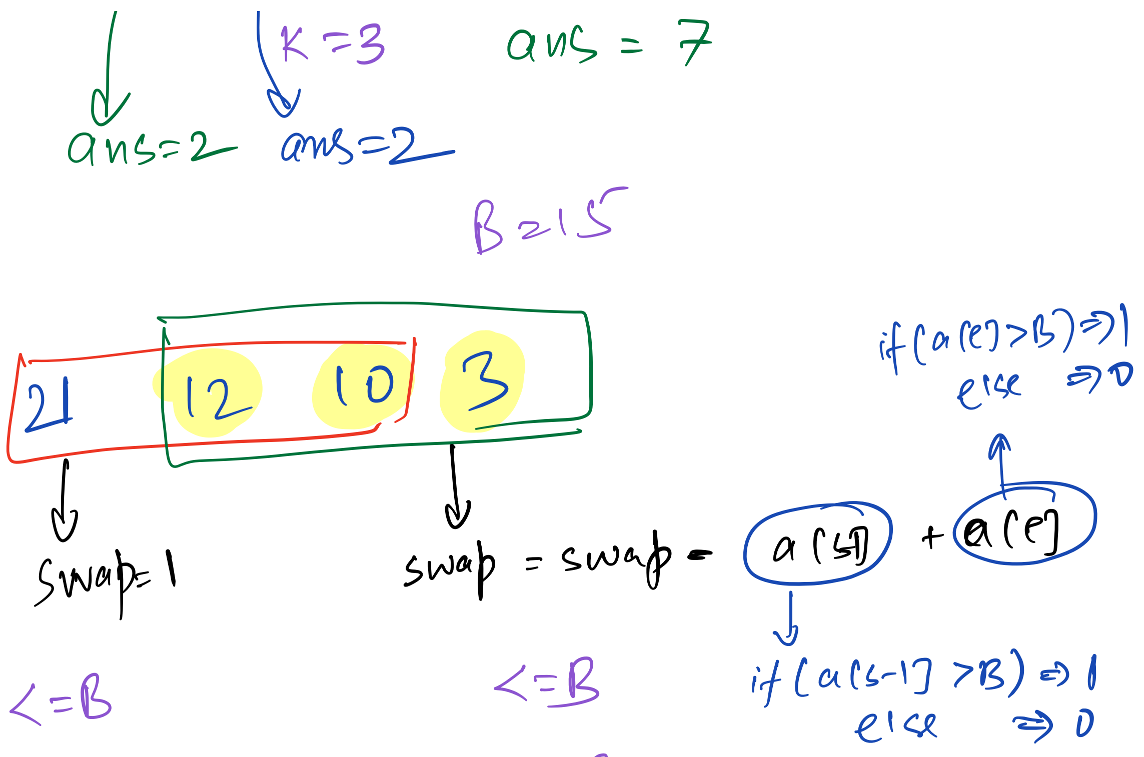
```
int minSwaps (int a[], int B) {
    n = a.length
    K = 0
    for (i = 0; i < n; ++i) {
        if (a[i] <= B)
            ++K;
    }
    int s = 0, e = K-1;        ans = n
    while (e < n ) {   or  s <= n-K
      swap = 0;
        for (i = s;  i <= e; ++i ) {
            if (a[i] > B)  ++swap;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

TC: $O(N^2)$

SC: $O(1)$

eg :





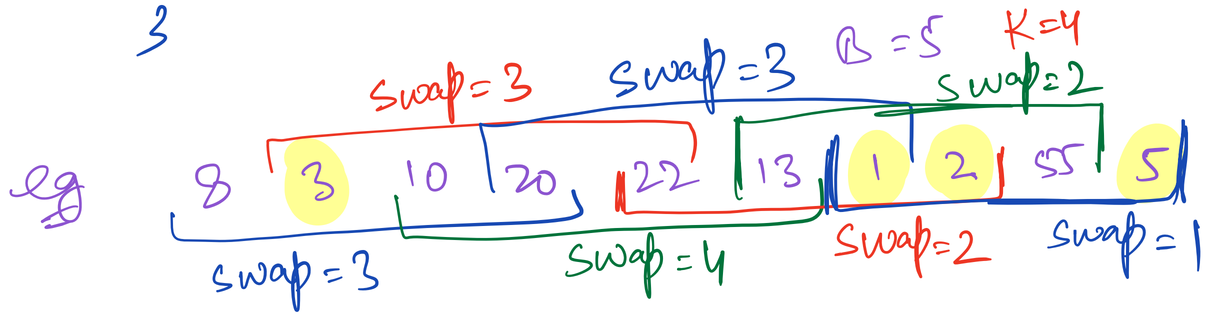
diff=0	<=B	<=B
+1	<=B	>B
-1	>B	<=B
0	>B	>B

```
int minSwaps(int a[], int 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;
```

TC: O(N)

SC: O(1)

```
s=1 , e=K
while (e<n) {
    // get subarray swap from [s,e]
    if( a[s-1] > B) --swap;
    if (a[e] > B) ++swap;
    if(ans > swap) ans=swap;
    s++ , e++;
}
return ans;
}
```



ans= ~~3~~ ~~2~~ 1