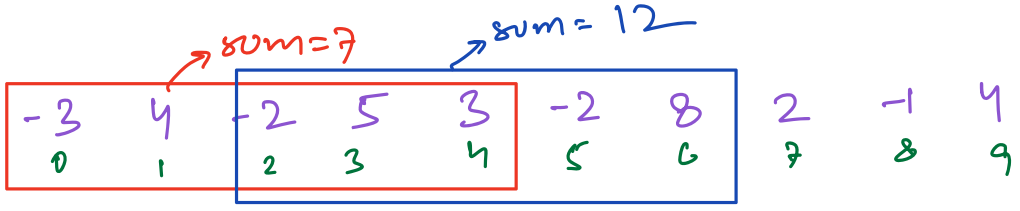
Arrays: Sliding Window

Question

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

eg a[10] =



K = 5

for first subarray of range [s, e]

s = 0

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

for last subarray of range [s, e]

$e = n - 1$

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

$\Rightarrow s = n - K$

Brute force : Traverse over all subarrays, find sums and then find max.

Code

```
def subarraySum(a[], K) {
    n = a.length
    s = 0, e = K-1
    ans = INT_MIN
    while(e<n) {
        sum = 0
        for(i=s; i<=e; ++i)
            sum += a[i]
        ans = max(ans, sum)
        s++, e++
    }
    print(ans)
}
```

n-K+1 iterations → while

K iterations → for

ans = 0 ✗ if all values are -ive

ans = a[0] ✗

a[3] = {100 -10 -10}, K=2

ans = sum of all values ✗

ans = min of array ✗

a[3] = {-1 -1 -1}, K=2

ans = INT_MIN ✓

↳ write in your own language

start index of first subarray = 0

start index of last subarray = n-K

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

= n-K+1

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

SC : O(1)

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

$K=1$ → $O(1*(n-1+1))$ = $O(n)$

$K=n$ → $O(n*(n-n+1))$ = $O(n)$

$K=n/2$ → $O(\frac{n}{2}*(n-\frac{n}{2}+1))$

$O(\frac{n}{2}*(\frac{n}{2}+1))$

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

$= O(n^2)$

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

⇓

$O(nK-K^2+K)$

$\Rightarrow O(n*K)$

TC: $O(N^2)$

SC: $O(1)$

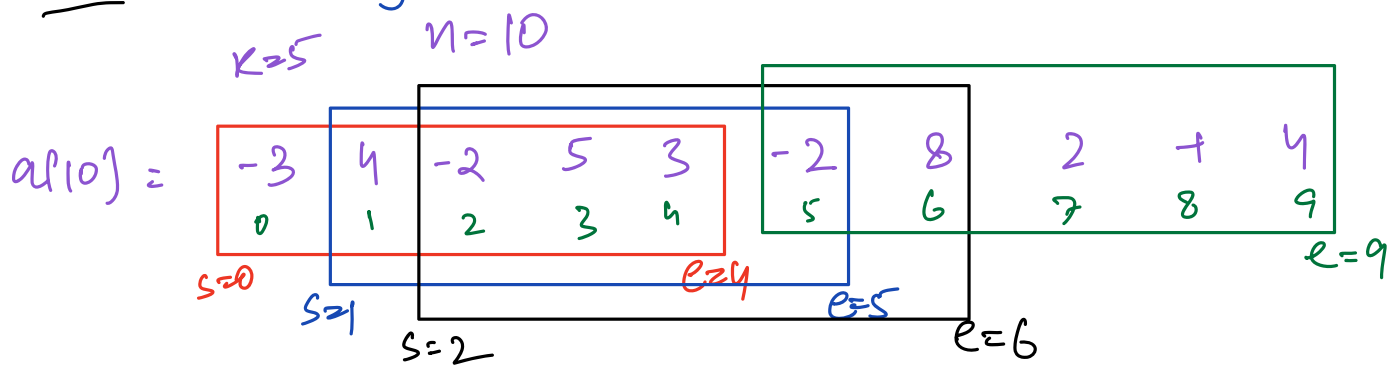
Ideal : Prefix Sum

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

SC : $O(N)$

] → TODO

Idea 2 : Carray forward aka Sliding Window



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

$sum_2 = sum_1 - (-3) + (-2) = 7 + 3 - 2 = 8$

$sum_3 = sum_2 - (4) + (8) = 8 - 4 + 8 = 12$

a[s-1] a[e]

Code

```
def subarraySum (a[], K) {
    n = a.length
    sum = 0
    for (i=0; i<K; ++i) {        } → K iteration
        sum += a[i]              } → sum of first subarray (sum1)
    }
    ans = sum
    S=1, e=K  [ e-s+1=K ]
    while (e<n) {                → n-K iteration
        sum = sum - a[s-1] + a[e]
        ans = max(ans, sum)
```

st++, en++

}

print(ans)

}

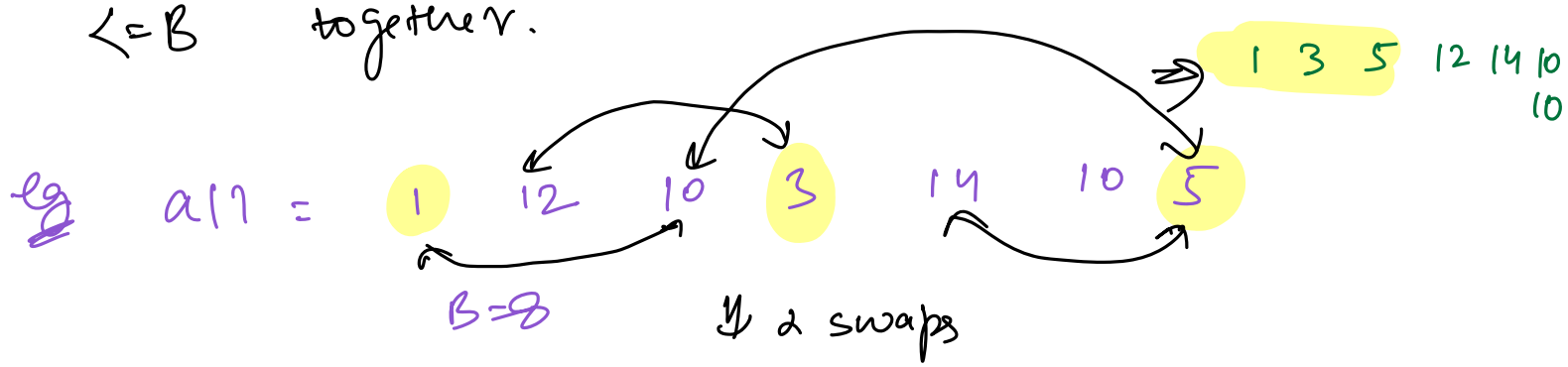
TC: $O(K + n/K)$

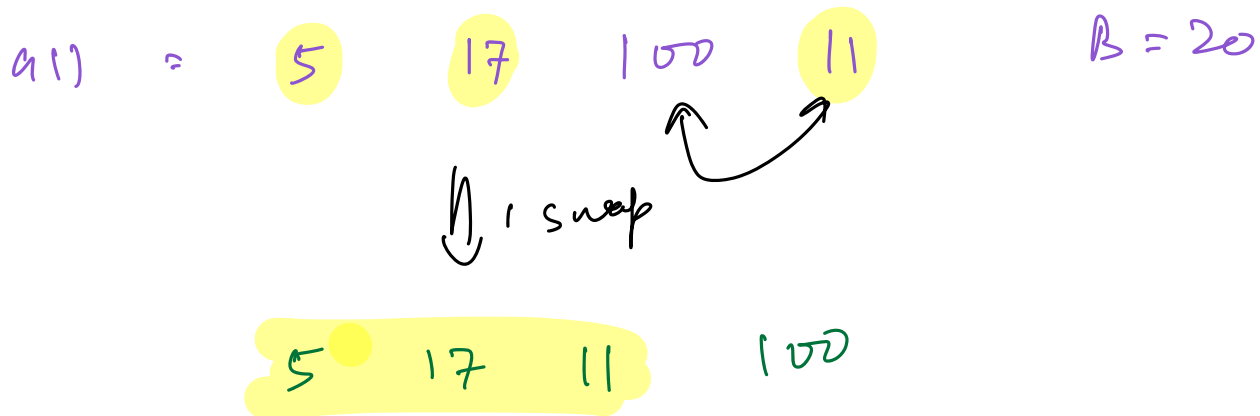
$\doteq O(N)$

SC: $O(1)$

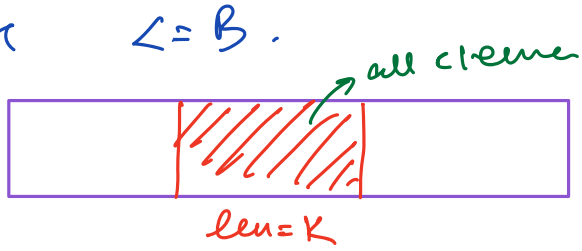
Question 2

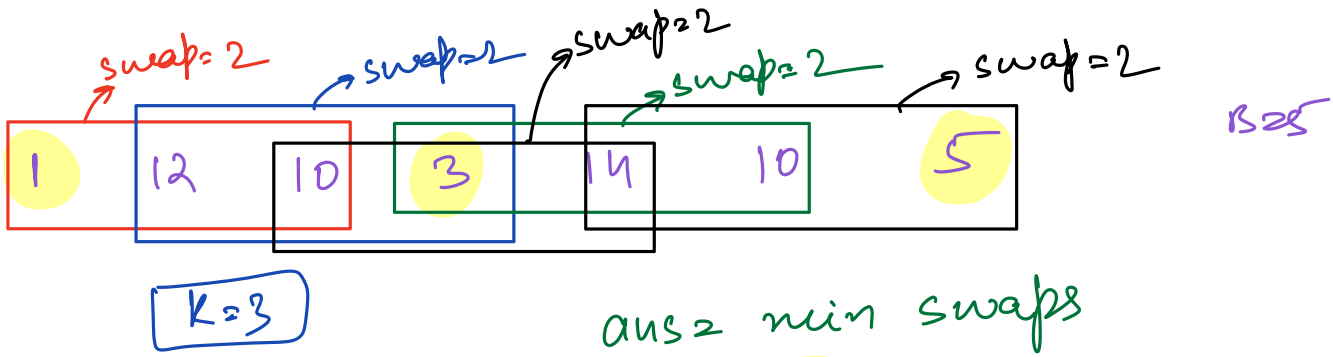
Given an array A and integer B, find minimum swaps required to bring all numbers <=B together.





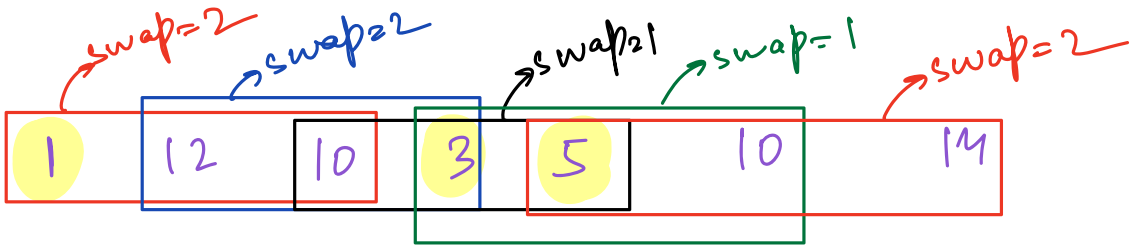
Idea : let say there are K elements which are <= B.





ans = min swaps

= 2



K=3

ans = 1

Code

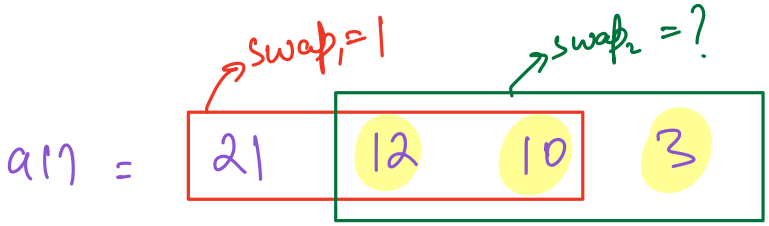
```
def minSwaps (a[], B) {
    n = a.length
    K = 0  → size of subarray

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

→ O(N)

```
s=0, e=k-1 , ans=K
while(e<n){ ⟶ n-k+1
    swap=0
    for(i=s; i<=e; ++i ){ ⟶ K
        if(a[i] > B)
            ++swap
    }
    ans = min(ans, swap)
    s++, e++
}
print(ans)
}
```

TC : $O(N^2)$

SC : $O(1)$



B=15

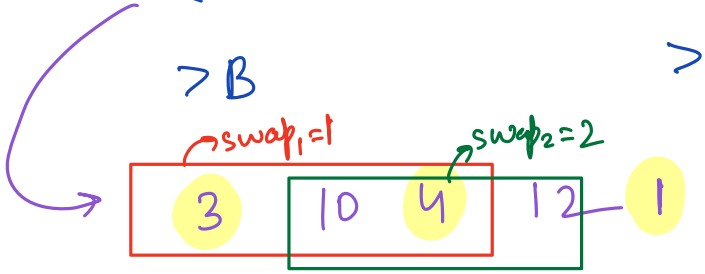
a[s-1]	a[e]
<=B	<=B
>B	<=B
<=B	>B
>B	>B

$swap_2 = swap_1$

$swap_2 = swap_1 - 1$

$swap_2 = swap_1 + 1$

$swap_2 = swap_1$



B = 6

Code

```
def 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) {
        if (a[s-1] > B)
            --swap
        if (a[e] > B)
            ++swap
        ans = min(ans, swap)
        s++, e++
    }
    print(ans)
}
```

→ O(N)

→ swap, → O(N)

→ n-K

a[s-1]	a[e]	same
<=B	<=B	
<=B	>B	+1
>B	<=B	-1
>B	>B	-1+1=0

TC: O(N)

SC: O(1)

Doubts

Print the spiral matrix

int a[10^{10}] → MLE error

$1 <= N <= 10^6$ → int a[N]