07. Sliding Window

14:08

1. Introduction
2. Problems

Introduction

- Sliding Window is a concept used to solve array problems.
- Solution approach is combination of carry forward technique and all array window of same size
  carry Forward + all array window of same size = Sliding Window.

Ex :

X * X X X X ✓ ✓ ✓ ✓
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

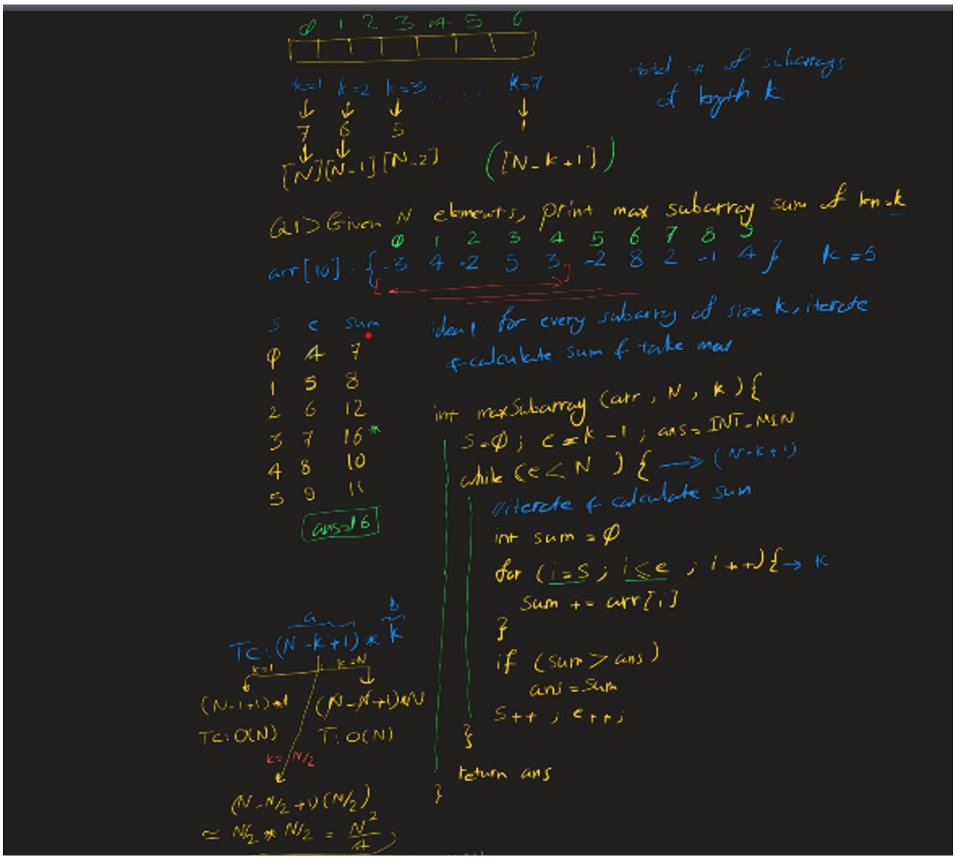
max = ~~15~~ ~~21~~ 27 . . .

Find Subarray with maximum size of window 5

① Calculate sum of 1st 5 element
② Keep adding/removing ele to get maxm sum

Problems

1. Given an array **A** of length **N**. Also given are integers **B** and **C**.
   Return 1 if there exists a subarray with length **B** having sum **C** and 0 otherwise
   Idea 1 :



0 1 2 3 4 5 6

k=1 k=2 k=3 . . . . k=7
↓ ↓ ↓ ↓
7 6 5 1
↓ ↓
[N] [N-1] [N-2] ([N-k+1])

total # of subarrays of length k

Q1> Given N elements, print max subarray sum of len=k

	0	1	2	3	4	5	6	7	8	9
arr[10] :	-3	4	-2	5	3	-2	8	2	-1	4

k = 5

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

ans=16

idea 1: for every subarray of size k, iterate & calculate sum & take max

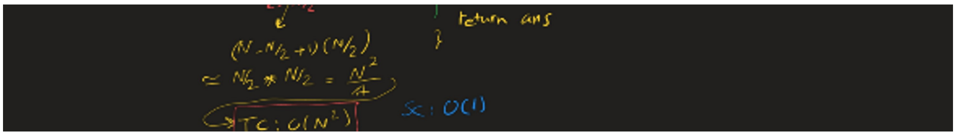
```
int maxSubarray (arr, N, k){
    s=0; e=k-1; ans=INT_MIN
    while (e<N ){ ⟶ (N-k+1)
        //iterate & calculate sum
        int sum = 0
        for (i=s; i≤e; i++){ → k
            sum += arr[i]
        }
        if (sum > ans)
            ans = sum
        s++; e++;
    }
    return ans
}
```

$TC: \underbrace{(N-k+1)}_{a} * \underbrace{k}_{b}$

k=1: $(N-1+1)*1$, $TC: O(N)$

k=N: $(N-N+1)*N$, $T: O(N)$

k=N/2: $(N-N/2+1)(N/2)$
$\simeq N/2 * N/2 = \frac{N^2}{4}$



$(N - N/2 + 1)(N/2)$

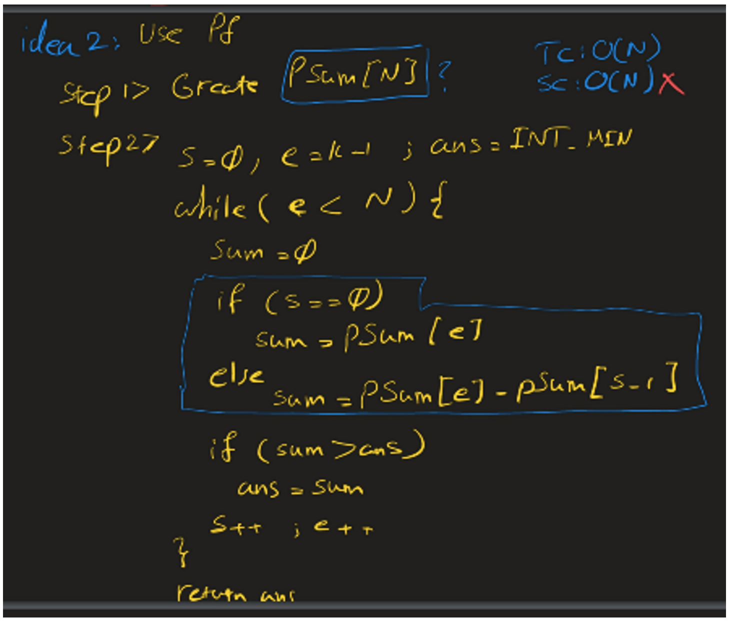
$\simeq N/2 * N/2 = \frac{N^2}{4}$

$\boxed{TC: O(N^2)}$

```
    return ans
}
```

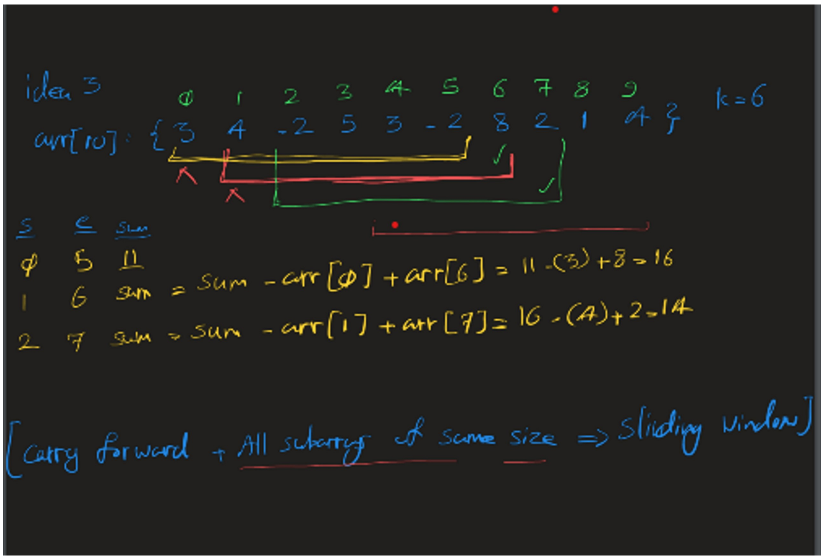
SC: O(1)

Idea 2 :



```
idea 2: Use Pf                                  TC: O(N)
  Step 1> Create [PSum[N]] ?                    SC: O(N)
  Step2> s = 0, e = k-1 ; ans = INT_MIN
         while ( e < N ) {
            Sum = 0
            if (s == 0)
               sum = PSum [e]
            else
               sum = PSum[e] - pSum[s-1]
            if (sum > ans)
               ans = sum
            s++ ; e++
         }
         return ans
```

Idea3 :



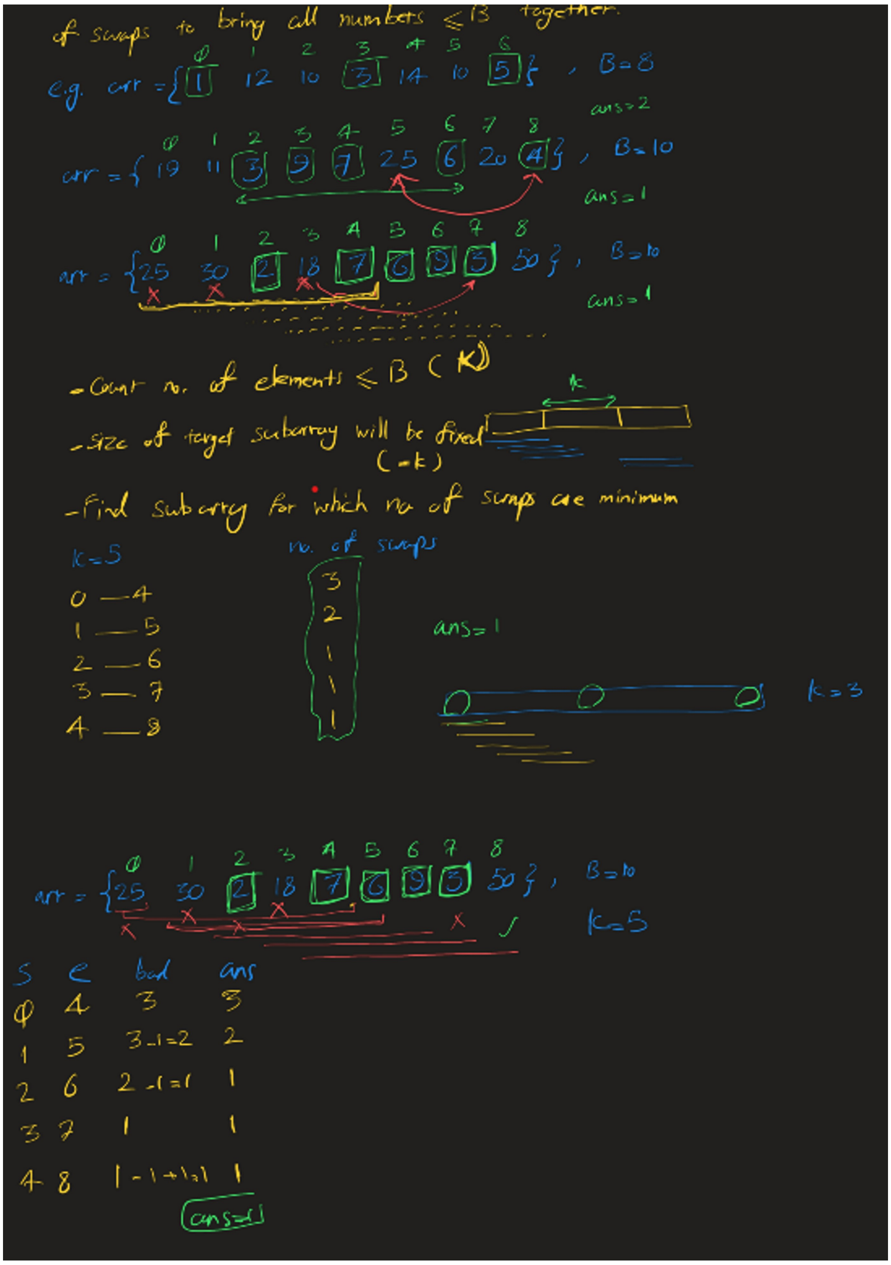
idea 3

	0	1	2	3	4	5	6	7	8	9	
arr[10]: {	3	4	-2	5	3	-2	8	2	1	4 }	k = 6

s	e	sum
0	5	11
1	6	sum = sum - arr[0] + arr[6] = 11 - (3) + 8 = 16
2	7	sum = sum - arr[1] + arr[7] = 16 - (4) + 2 = 14

[Carry forward + All subarrys of same size => Sliding window]

2. Given an array of integers **A** and an integer **B**, find and return the minimum number of swaps required to bring all the numbers less than or equal to **B** together.
   Note: It is possible to swap any two elements, not necessarily consecutive.



of swaps to bring all numbers ≤ B together.

e.g. arr = { $\boxed{1}$ 12 10 $\boxed{3}$ 14 10 $\boxed{5}$ } , B = 8

ans = 2

arr = { 19 11 $\boxed{3}$ $\boxed{9}$ $\boxed{7}$ 25 $\boxed{6}$ 20 $\boxed{4}$ } , B = 10

ans = 1

arr = { 25 30 $\boxed{2}$ 18 $\boxed{7}$ $\boxed{6}$ $\boxed{9}$ $\boxed{3}$ 50 } , B = 10

ans = 1

- Count no. of elements ≤ B (K)
- Size of target subarray will be fixed (= k)
- Find subarray for which no of swaps are minimum

k = 5

0 — 4
1 — 5
2 — 6
3 — 7
4 — 8

no. of swaps

3
2
1
1
1

ans = 1

k = 3

arr = { 25 30 $\boxed{2}$ 18 $\boxed{7}$ $\boxed{6}$ $\boxed{9}$ $\boxed{3}$ 50 } , B = 10

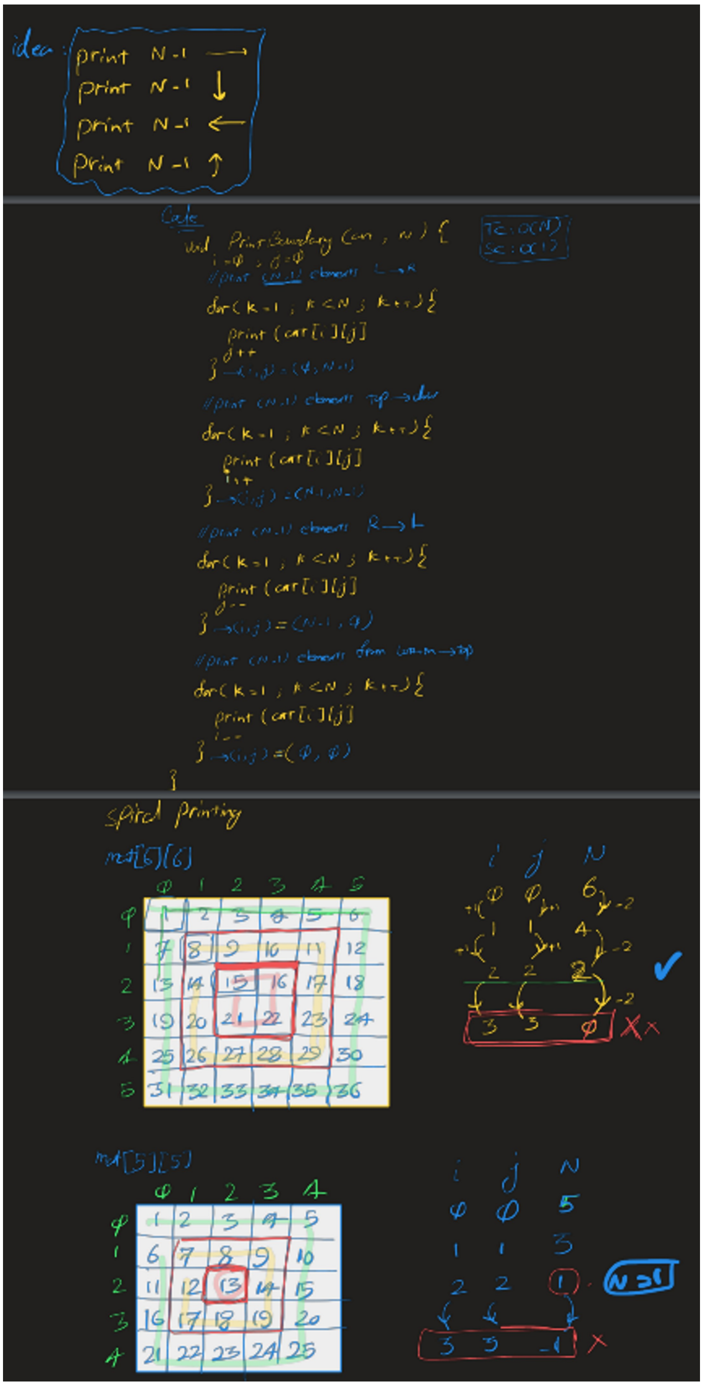
k = 5

s	e	bad	ans
∅	4	3	3
1	5	3 - 1 = 2	2
2	6	2 - 1 = 1	1
3	7	1	1
4	8	1 - 1 + 1 = 1	1

$\boxed{ans = 1}$

3. **Given an integer A, generate a square matrix filled with elements from 1 to A2 in spiral order and return the generated square matrix.**

4.



idea :

```
print N-1 →
print N-1 ↓
print N-1 ←
print N-1 ↑
```

Code

```
void PrintBoundary (arr , N ) {
    i = φ ; j = φ
    //print (N-1) elements L→R
    for( k=1 ; k<N ; k++){
        print (arr[i][j]
        j++
    } → (i,j) = (φ, N-1)
    //print (N-1) elements top → down
    for( k=1 ; k<N ; k++){
        print (arr[i][j]
        i++
    } → (i,j) = (N-1, N-1)
    //print (N-1) elements R → L
    for( k=1 ; k<N ; k++){
        print (arr[i][j]
        j--
    } → (i,j) = (N-1 , φ)
    //print (N-1) elements from bottom → top
    for( k=1 ; k<N ; k++){
        print (arr[i][j]
        i--
    } → (i,j) = ( φ , φ )
}
```

TC : O(N)
SC : O(1)

Spiral Printing

mat[6][6]

	φ	1	2	3	4	5
φ	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

i	j	N
φ	φ	6
1	1	4
2	2	2
3	3	φ

+1 +1 -2 ✓ ✗✗

mat[5][5]

	φ	1	2	3	4
φ	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

i	j	N
φ	φ	5
1	1	3
2	2	1
3	3	-1

N≥1 ✗

5. Given an array **A** of size **N**, find the subarray of size **B** with the least average.