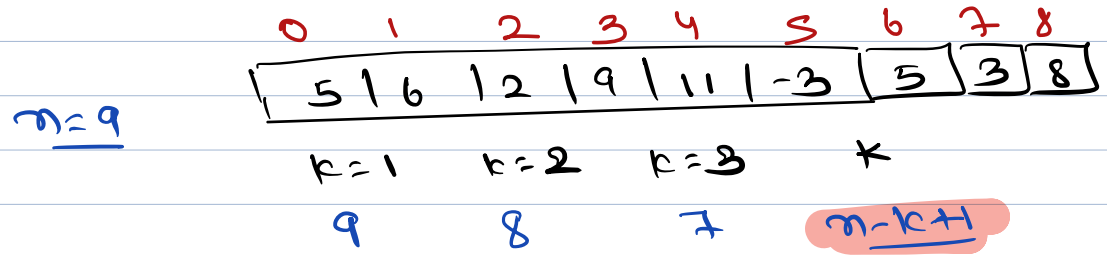
Today's Content

Sliding window
2-3 problems on it.

Problem Solving -> Sunday - 2pm.
↳ optional class
↳ recordings
→ least submitted problems till now.

Today's Quote :-

Win the game so you can free of it. → @Naval Ravikant



Ques) Given N elements print max subarray sum of length = 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

Subarrays

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

Ans = 16

idea :- for every subarray of size k, iterate & calc sum. Overall max sum will be our ans.

```
int maxSubarray(arr, n, k){
    s = 0, e = k-1, ans = -∞
    while (e<n)
        int sum=0
        for(i=s; i<=e; i++){
            sum += arr[i]
        }
        if(sum>ans){ans=sum}
        s++, e++
    return ans
}
```

T.C → (n-k+1) * k

k = 1 , k = n

when k = 1 : (n-1+1) * 1 = n

when k = n : (n-n+1) * n = n

when k = n/2 : (n - n/2 + 1) n/2 = (n/2 + 1)(n/2)

Worst → O(n²)

S.C → O(1)

idea 2 :-

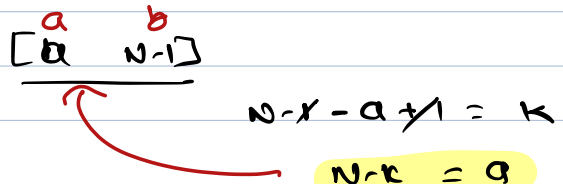
① // create prefix sum

② s = 0, e = k-1, ans = -∞

```
while (e<n)
    int sum=0
    if (s==0) sum = pfsum[e]
      else   sum = pfsum[e] - pfsum[s-1]
    if (sum>ans) {ans=sum}
    s++, e++
}
return ans
```

T.C → O(n)

S.C → O(n)



idea 3 :-

k = 6

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

s	e	sum
0	5	11
1	6	sum + arr[6] − arr[0] = 16
2	7	sum + arr[7] − arr[1] = 14
3	8	sum + arr[8] − arr[2] = 17
4	9	sum + arr[9] − arr[3] = 16

17 ans

(s-1 e-1)

s e sum + arr[e] − arr[s-1]

Carry forward + All subarrays of same size → sliding window

final code :-

```
int maxSum(arr, N, k) {
    1:- calculate the sum of first k elements
    sum = 0
    for (i=0; i<k; i++) {
        sum += arr[i]
    }                                   ] k
    ans = sum, s = 1, e = k
    while (e < N) {                     ] N-k
        // calculate sum of subarray [s-e]
        sum = sum + arr[e] - arr[s-1]
```

if (sum > ans) { ans = sum }
s++, e++

return ans;

T.C → O(N), S.C → O(1)

Ques) Given arr[N] and a number B.
Find and return min no of swaps to bring all numbers <= B together.

e.g.

arr = { 1 12 10 3 14 10 5 3 }, B = 8

0 1 2 3 4 5 6

Ans = 2.

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

Ans = 1.

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

Ans = 1

Obs:- 1) Count of all elements <= B [k]
→ size of subarray is fixed k

→ find subarray for which no of swaps are minimum.

	0	1	2	3	4	5	6	7	8	
arr = {	25	30	2	18	7	6	9	3	503	}, B = 10

Good element <= B
Bad element > B.

k = 5.

Subarray	Bad elements
0 - 4	3
1 - 5	2
2 - 6	1
3 - 7	1
4 - 8	1

Pseudocode :-

```
int minSwaps(arr, n, B)
    1) count no. <= B
    k = 0
    for(i=0; i<n; i++){
        if(arr[i] <= B) { k++;
    }
                    // k == n
    if(k==0 || k==1) { return 0;

    // Calc. no. of Bad elements for
       first window.
    bad = 0;
```

```
for (i=0; i<k; i++) {
    if (arr[i] > B) {
        bad++
    }
}

// Apply sliding window Technique
ans = bad, s=1, e=k
while (e<n) {
    if (arr[s-1] > B) { bad--; }
    if (arr[e] > B) { bad++; }
    if (bad < ans) { ans = bad; }
    s++, e++
}

return ans;
}
```

T.C → O(n)
S.C → O(1)

Break 10:10 - 10:20 pm

Ques) find no. of triplets i, j, k, such that, $i<j<k$ && $arr[i] < arr[j] < arr[k]$

	0	1	2	3	4
Eg:-	3	4	6	9	2

Triplets :-

i, j, k
0 1 2
0, 2, 3
0, 1, 3
1, 2, 3

→ 4 only

	0	1	2	3	4
Eg:-	2	6	9	4	10

Triplets :-

0, 1, 2
0, 1, 4
0, 2, 4
1, 2, 4
0, 3, 4

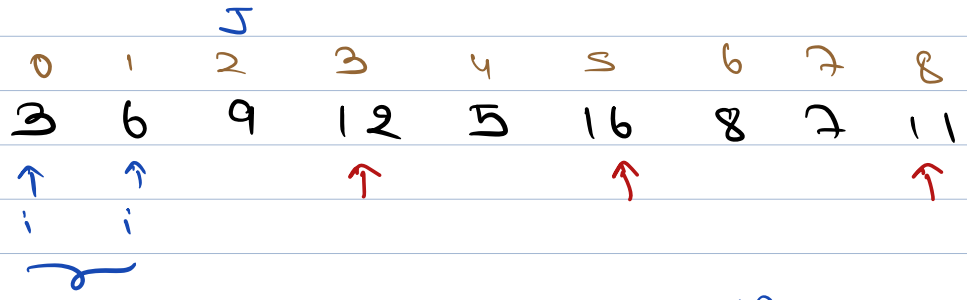
} → 5 only

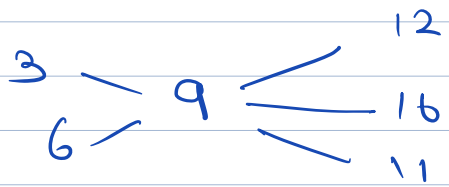
Brute force :-

```
count = 0;
for (i=0; i<n; i++) {
    for (j=i+1; j<n; j++) {
        for (k=j+1; k<n; k++) {
            if (arr[i]<arr[j]<arr[k]){
                count++
            }
        }
    }
}
return count.
```

T.C → $O(n^3)$
S.C → $O(1)$.

Eg:- i, j, k





⇒ 6 triplets

for every j, L * R.

no. of elements less than arr[j] in 0 to j-1 → L

no. of elements greater than arr[j] in j+1 to n-1 → R

```
cnt = 0;
for (j = 1; j < n-1; j++) {
    // j is middle element
    left = 0;
    for (i = j-1; i >= 0; i--) {
        arr[i] < arr[j]) {
            left++
        }
```

3

```
right = 0;
for (i = j+1; i<n; i++) {
    if (arr[i] > arr[j]) {
        right++
    }
}
cnt = cnt + (left * right);
```

prefix is not possible ✗

T.C → $O(n^2)$
S.C → $O(1)$

T.C → $O(n \log n)$ → Balanced BST
Segment Tree.

Ques) Product array Puzzle

Given n array elements
replace every element.
→ arr[i] with product of all elements except itself.
→ can't use / in your code.

Eg 1) 9, 4, 1, 3, 5
→ 60 30 40 24
120

Ex 2) 1, 6, 2, 3 → 36

36 6 18 12

9, 4, 1, 3, 5

pf product = 2, 8, 8, 24, 120

sf product = 120, 60, 15, 15, 5

Prod of all elements except me at ith

idx = pf product[i-1] * sf prod[i+1]

Pseudo code :-

// pf product (T.C → O(N))

// sf product (S.C → O(N))

```
for (i=1; i<N-1; i++) {
    ans[i] = pfprod[i-1] * sfprod[i+1]
}

ans[0] = sfprod[1]
ans[n-1] = pfprod[n-2]
```

2, 4, 1, 3, 5

60 30

If product = 120, 60, 15, 15, 5

arr[][]

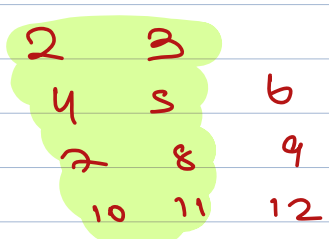
for (i=0; i< arr.length; i++) {

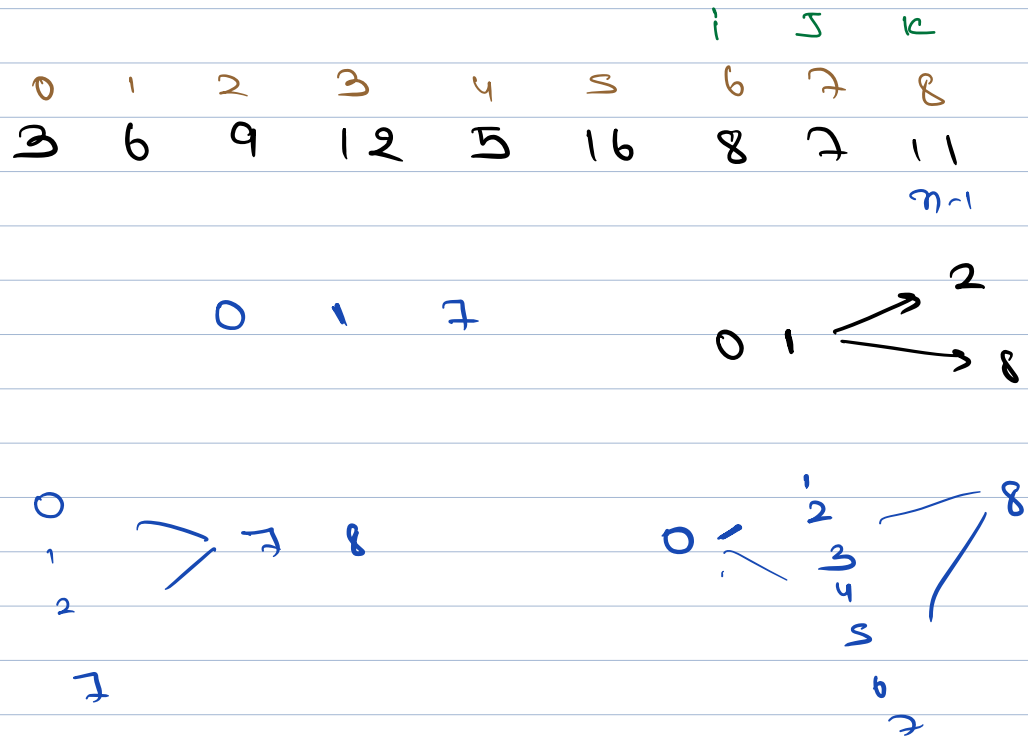
for (j=0; j < arr[i].length; j++)

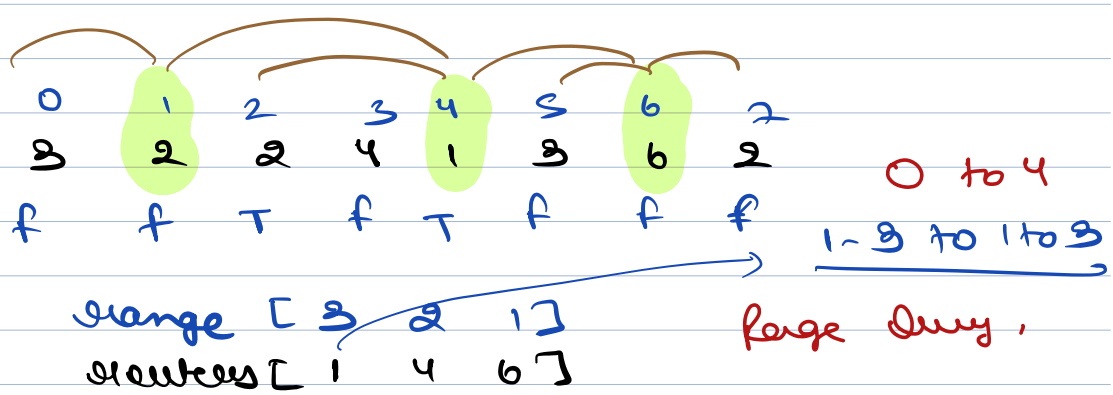
{

}

}



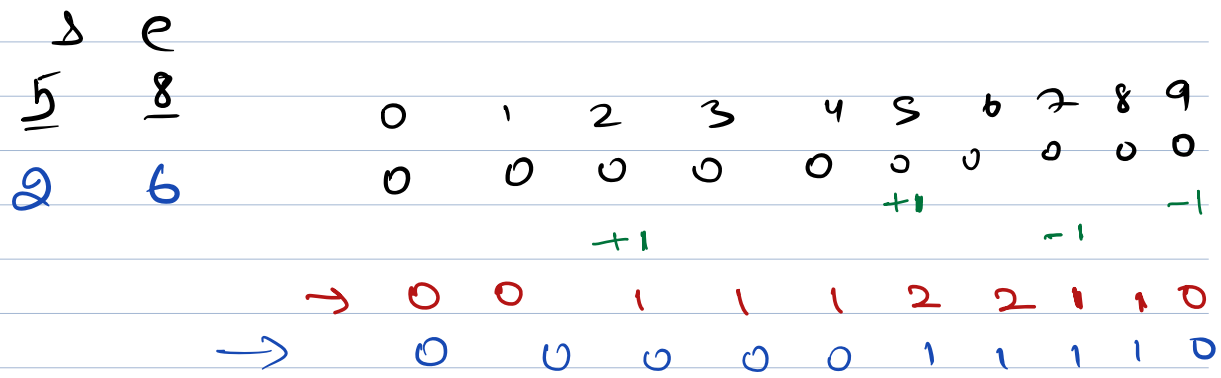

i j k
0 1 2 3 4 5 6 7 8
3 6 9 12 5 16 8 7 11
n-1
0 1 7
0 1 2 8
0 1 2 7 8
0 1 2 3 4 5 6 7
3 5 6

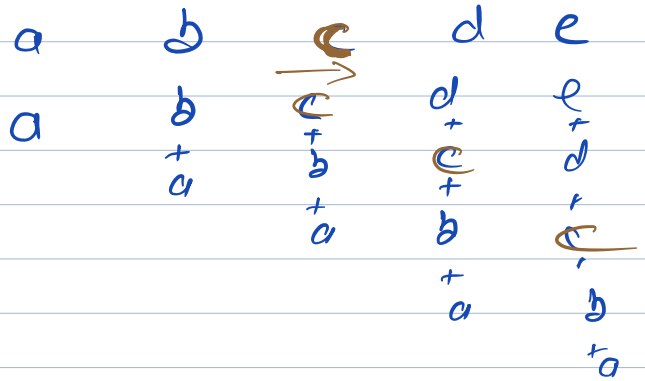

0 1 2 3 4 5 6 7
3 2 2 4 1 3 6 2
f f T f T f f f
0 to 4
1-3 to 1 to 3
range [3 2 1]
queries [1 4 6]
large Query.
3 2 2 4 1 3 6 2
+1 +1 -1 -1
0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7
1 1 2 2 2 2 2 1

4 → 2

2 to 6

5 to 7


5 8
2 6
0 1 2 3 4 5 6 7 8 9
+1
+1
-1
-1
→ 0 0 1 1 1 2 2 1 1 0
→ 0 0 0 0 0 1 1 1 1 0


a b c d e
a
b a
c b a
d c b a
e d c b a