Welcome ☺

Agenda : 1 ques from Prefix Sum
Carry forward
Subarrays
Problems

Q Given an array of size N, count number of special index in the array.

Special Index : Those indexes which if removed, sum of all even indexed elements is equal to sum of all odd indexed elements.

eg: A: [4 3 2 7 6 -2]
(0 1 2 3 4 5)

i	A[]	Se	So	
0	[3 2 7 6 -2] (0 1 2 3 4)	8	8	✓
1	[4 2 7 6 -2] (0 1 2 3 4)	9	8	✗
2	[4 3 7 6 -2] (0 1 2 3 4)	9	9	✓
3		4	9	✗
4		4	10	✗
5		12	10	✗

Obs 1 ⇒ Index of element only after removed index are going to be affected

Sum of even indexed elements $\Rightarrow Se(0 \to i-1) + So(i+1, N-1)$

Sum of odd indexed elements $\Rightarrow So(0 \to i-1) + Se(i+1, N-1)$

1) Create prefix sum for even & odd indexed elements $O(N+N)$ / $O(N)$
2) Run a loop from $0 \to N-1$ $O(N)$
   1) Remove index i & compare So & Se $O(1)$
   2) Whenever $So == Se$, increment count $O(1)$

T.C $\Rightarrow$ $O(N+N+N) \simeq O(N)$

S.C $\Rightarrow$ $O(N+N) \simeq O(N)$

H.W

Q Given a string s of lowercase characters, return the count of pairs (i, j) s.t $i<j$ & $s[i] == 'a'$ and $s[j] == 'g'$

count = 3

eg: s = "abegag"

Brute force: For every 'a', we need to find count of 'g's on right side of a. So, we need to have nested loops.

T.C $\to N^2$
S.C $\to O(1)$

eg:

0	1	2	3	4	5	6	7	8
a	c	b	a	g	K	a	g	g
1	1	1	2	2	2	3	3	3

count = ~~0~~ ~~7~~ ~~5~~ 8

TC ⇒ $O(N+N) = O(N)$ ~~$O(N+N)$~~ ×

SC ⇒ ~~$O(N)$~~ $O(1)$

Code

```
result = 0
count_a = 0
for ( i -> 0 to N-1)
{
    if ( str[i] == 'a' ) {
            count_a ++
    }
    else if ( str[i] == 'g' ) {
            result += count_a
    }
}
return result
```

Subarrays

→ Contiguous part of an array.

→ Can have one or more elements

Order ✓
Contiguous ✓

eg: 4 1 2 3 -1 6 9 8 12

⇒ 2 3 -1 6 ✓

⇒ -1 3 2 ×

⇒ 9 ✓

⇒ 4 1 2 3 -1 6 9 8 12 ✓

⇒ 4 12 ×

⇒ 1 2 3 6 ×

⇒ Represent a subarray

1) Start & End Index of subarray
2) Start index & length of subarray.

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

sum = 7

subarrays in an array ⇒ $N + N-1 + N-2 + N-3 + \cdots 1$

$\Rightarrow \frac{N(N+1)}{2}$

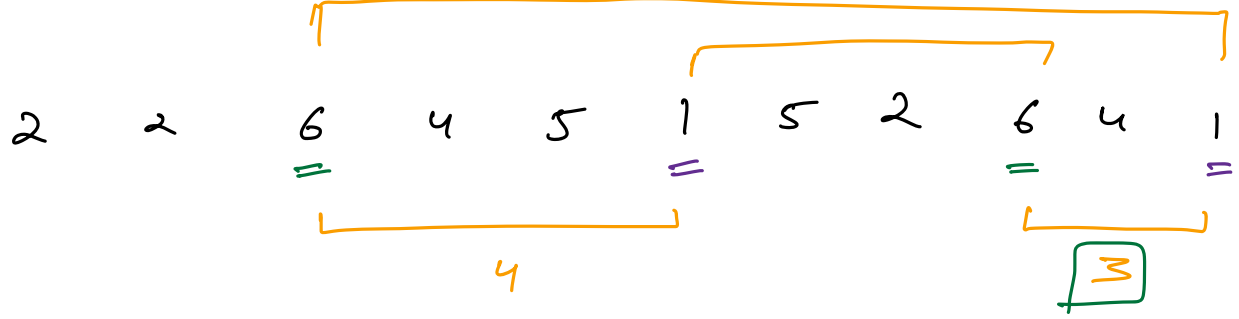
Q Print all possible subarrays

```
for( i -> 0 to N-1)
{
    for( j -> i to N-1)
    {
        print( ) → O(N)
    }
}
```

T.C ⇒ $O(N^3)$

i	j	print
0→N-1	i→N-1	

Q Given an array of N integers, return length of smallest subarray which contains both maximum & minimum element of the array.



Ans = 3

Bruteforce Check for all subarrays

T.C ⇒ ~~O(N³)~~ ⇒ O(N²)

S.C ⇒ O(1)

Optimized

N.A

Next Class

⇒ Sliding Window.