1Q) Given N Array elements replace every element
→ ar[i] with prod of all array elements except itself
→ You cannot / in your code?

	0	1	2	3	4
Ex1: ar[5] =	{ 2,	4,	1,	3,	5 }
ar[5] =	{ 60,	30,	120,	40,	24 }

	0	1	2	3	
Ex2: ar[4] =	{ 1	6	2	3 }	Prod = 36
ar[4] =	{ 36/1,	36/6,	36/2,	36/3 }	

	0	1	2	3	4	5	6
Ex3: ar[7] =	{ 2	1,	3,	2,	1,	4,	2 }

Note: Prefix : Starting at 0^{th} index

$\boxed{pf[i] : \text{product of all elements } [0\_i]}$

Suffix : Ending at $N\text{-}1^{th}$ index

$\boxed{sf[i] = \text{product of all elements } [i, N\text{-}1]}$

Pseudo Code

```
// prod[N]

i = 0; i < N; i++) {

    i } -> product of all Elements Except i-th index
           prod[0, i-1] * prod[i+1, N-1]
           Prep[i-1] : [0, i-1]   Sufp[i+1] : [i+1, N-1]

    left = 1  }
    j = 0; j < i; j++) { left = left * ar[j] }

    right = 1
    j = i+1; j < N; j++) { right = right * ar[j] }

    prod[i] = left * right

}
```

TC: N * [N]

TC: O(N^2)

Idea: We use Prefix product

$\boxed{prep[i] = \text{Product of all Elements from } [0-i]}$

$prep[i] = \underbrace{prep[i-1]}_{[0,\ i-1]} * ar[i]$

$\underbrace{[0,\ i-1] * ar[i]}$ } $\Rightarrow [0, i]$

Note: Product of all indices from $[i - j]$ =

$prod \Rightarrow [\underbrace{0, 1, 2, .. i-1}, i, .. j]$

$Prep[j] = Prep[i-1] * prod[i-j]$

$$\boxed{Prod[i-j] = \frac{Prep[j]}{Prep[i-1]}}$$

Psudo Code

Prep[N]:

```
Prep[0] = ar[0]
i = 1; i < N; i++) {
    Prep[i] = Prep[i-1] * ar[i]
}
```

Sufp[N]: Sufp[i] = prod of all [i - N-1]

```
sufp[N-1] = ar[N-1]
i = N-2; i >= 0; i--) {
    Sufp[i] = ar[i] * sufp[i+1]
}

i = 0; i < N; i++) {
    left = prod[0, i-1] = prep[i-1]   } if i == 0
                                        left = 1
    right = prod[i+1, N-1] = sufp[i+1] } if i == N-1
                                        right = 1
    ar[i] = left * right
}
```

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

$\to O(N)$

TODO:

1) Carry forward your left product

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

28) Max Consecutive 1's

[Every arr[i] is 0 or 1]

Given a Binary array[], we can atmost replace a single 0 with 1 find max consecutive 1's we can get

Ex1: 0 1 1 1 0 1 1 0 1 1 0

4   3   3+2+1 = 6   2+2+1 = 5   2+0+1 = 3   2

ans = 6

Ex2: 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1

3   4   4   2   2   0   3   3   3   2   2

3+4+1 = 8

4+2+1 = 7

2+1+0 = 3

0+1+3 = 4

3+2+1 = 6

ans = 8

Ex3: 1 1 1 0 1 1 0 1 1 1 0 1

Ex4: 1 1 1 1 1 1   ans = 6

Sol:

Approach 1:

ans = 0

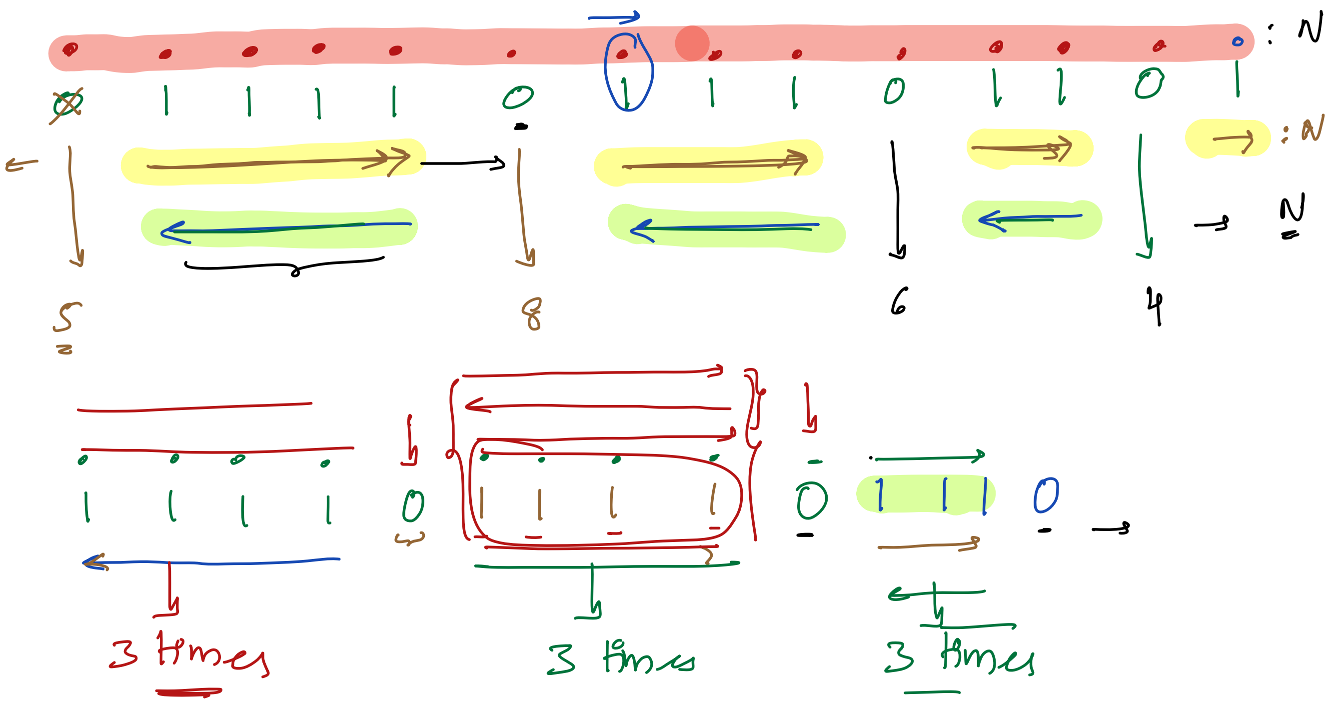
iterate & get cnt of 1's = cnt;

if (cnt == N) { return N }

```
i = 0; i < N; i++) {                          → TC: O(N)
    if ( arr[i] == 0 ) {
        l = 0; → { No: of consecutive 1's on left }
        j = i-1; j >= 0; j--) {
            if ( ar[j] == 1) { l++ }          → N
            else { break }
        }
        r = 0; → { No: of Consecutive 1's on right }
        j = i+1; j < N; j++) {
            if (ar[j] == 1) { r++ }
            else { break }
        }
        T = l + r + 1
        ans = max(ans, l+r+1)
    }
}
```

```
j = 1; j <= 3; j++) {
    i = 0; i < N; i++)
        print Car[i]
    }
}
```

→ Every dc = 3 times



Total iterat = Total inner loop iterations

1 should be with in array

Given a Binary array[], we can atmost swap a single 0 with 1 find max consecutive 1's we can get

Ex1: ar[7] : 1 1 ~~0~~ 1 1 ~~0~~ 1

2 1 2

4

ans = 5

Ex2: ar[9] = 1 1 ~~0~~ 1 0 1 0 1 1

2 + 1 + 1

Ex4: ar[7] : 1 1 1 0 1 1 1 → 1 1 1 1 1 1 0

left = 3 right = 3

Total ans = left + right

Ex5: ar[8] : 1 1 0 1 1 1 0 0

left = 2, right = 3

Total ans = 2 + 3 = 5

Approach 1:

ans = 0

iterate & get cnt of 1's = cnt;

if (cnt == N) { return N } ✓

```
i = 0; i < N; i++) {                                    } → TC: O(N)
    if (arr[i] == 0) {
        l = 0; → { No: of consecutive 1's on left }
        j = i-1; j >= 0; j--) {
            if ( ar[j] == 1) { l++ }                    } → N
            else { break }
        }
        r = 0; → { No: of Consecutive 1's on right }
        j = i+1; j < N; j++) {
            if (ar[j] == 1) { r++ }
            else { break }
        }
        if (cnt > l+r) { // I can spare extra 1
            ans = max (ans, l+r+1)
        }
        else {
            ans = max (ans, l+r)
        }
    }
}
```

38) Find no of triplets (i, j, k) such that

index

$\{ i < j < k$ && $ar[i] < ar[j] < ar[k]$

Ex 1

	0	1	2	3	4
Ex1	3	4	6	9	2

Triplets:

i	j	k	ar[i]		ar[j]		ar[k]	
0	1	2	3	<	4	<	6	✓
0	2	3	3	<	6	<	9	✓
0	1	3	3	<	4	<	9	✓
1	2	3	4	<	6	<	9	✓

ar[5] = {2, 6, 9, 4, 10}
(indices 0 1 2 3 4)

i	j	k	ar[i]		ar[j]		ar[k]	
0	1	2	2	<	6	<	9	✓
0	1	4	2	<	6	<	10	✓
0	2	4	2	<	9	<	10	✓
1	2	4	6	<	9	<	10	✓
0	3	4	2	<	4	<	10	✓

3 Nested loops

cnt = 0

```
i = 0; i < N; i++) {                    → O(N^3)   SC: O(1)

    j = i+1; j < N; j++) {
        if (ar[j] > ar[i]) {
            k = j+1; k < N; k++) {
                if( ar[i] < ar[j] < ar[k]) {
                    cnt++;
                }
            }
        }
    }
}
```

Can I say we are fixing 1st Element in triplet : ar[i]

→ Sunday : Problem Solving Assignment

: 2 – 4:30 PM

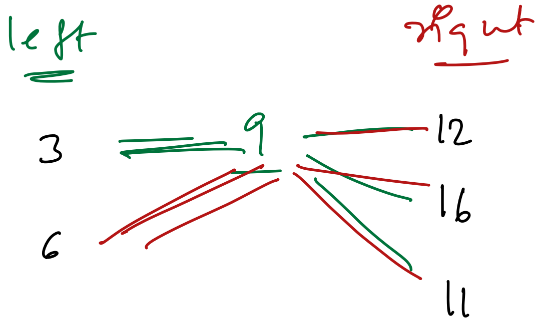
→ N bulbs ✓

→

Ex3:

	0	1	2	3	4	5	6	7	8
ar[] :	3	6	9	12	5	16	8	7	11
left :		1	2	3	1	5	3	3	
right :		6	3	1	4	0	1	1	

{Total Triplets} : 6 + 6 + 3 + 4 + 0 + 3 + 3 = 25



cnt = 0

i = 1 ; i < N-1 ; i++){

// ar[i]

left = number of less than ar[i] in left [0, i-1]

right = number of greater than ar[i] in right [i+1, N-1]

cnt = cnt + left * right

}

TC : O(N^2) SC : O(1)

→ Final Optimized Sol

→ TC : O(N log N)

1) Balanced BST

2) Segment Trees

arr[ ] : 3 4 8 7

→ 3 5 8 9 ~~6~~ 13 ~~8~~ 15

- 2