# Tentative Plan - 17th September 2021

Friday, 17 September 2021 5:24 PM

## Week - 17th September 2021 to 19th September 2021

### Day 1

1. ✓ No. of subarray with bounded max
2. ✓ Wiggle sort -1
3. ✓ wiggle sort -2
4. ✓ Range Addition
5. ✓ Product of array Except itself
6. Maximise distance to closest one

### Day 2

1. First Missing +ve
2. Best Meeting point
3. Max. Swap
4. ✓ Pascal's Triangle
5. 2 Sum
6. 3 Sum

### Day 3

1. 4 Sum
2. K Sum
3. complex No. multiplication
4. Min No. of plateform
5. Sieve of Eratosthenes
6. Two diff.

# Number of Subarray with bounded max

Friday, 17 September 2021 6:22 PM

Allowed time complexity → O(n) ← ×

Allowed space → O(1)

7

3 1 7 || 2

Brute →

1. Find all subarray and max
2. check that max is in Range or not
3. If max is in Range then count++; } Time → O(n²)

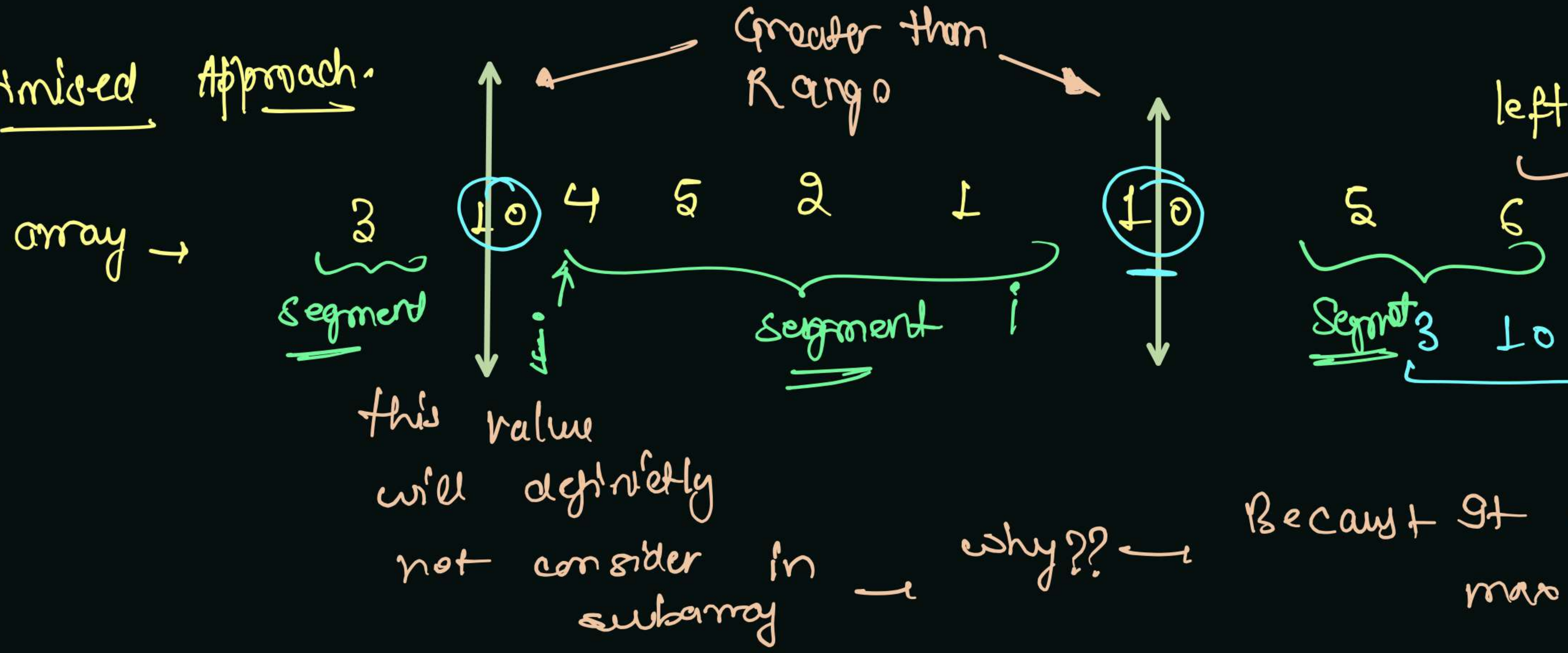
max = 6

2, 0, 1, 6 valid subarray

Range of max

left = 3, right = 7 ] Both are Included

optimised Approach.



Break point

3 10 2 5 — max 10

Invalid subarray

Because It create Impact on max of subarray.

## case-I , pointer. i , pointer j

left ≤ arr[i] ≤ right

arr[i] is in Range

arr → {3 1 4} 5

j ↑ (under 3), i ↑ (under 5)

3 5 0 1 1 2 4

j, i

i-j+1

no. of Subarray Ending at i

5
4 5
1 4 5
3 1 4 5

4
2 4
1 2 4
1 1 2 4
0 1 1 2 4
5 0 1 1 2 4
3 5 0 1 1 2 4

prev = i-j+1

Generic term ⇒ i-j+1

$\boxed{prev\_count = i-j+1}$ , count += (i-j+1);

## case-II   left=3 right=7

arr[i] < left

[3 5 0 1 1] 2 → arr[i] < left

j i i i

3
5
3 5

No. of Subarrays Ending at i

2 → max 2 ✗
1 2 → max 2 ✗
1 1 2 → max 2 ✗
0 1 1 2 → max 2 ✗ } Max is not in Range
5 0 1 1 2 → max 5 ✓
3 5 0 1 1 2 → max 5 ✓ } 2-count

last time answer is count
in Range

count += prev_count  from last in Range count

3 5  5
i j  3 5  prev_count=2

## case-III

right < arr[i]

[3 1 4] 10

j  i

No. of subarray Ending at 'i'

— } max → 10 out of Range

→ No impact on count  prev_count=0

$\boxed{j = i+1; \quad i++}$  Increment in 'i' and 'j'

Reset prev_count

coin

Refill

$$\begin{tikzcd} & \text{Refill} \arrow[dl] \arrow[dr] & \\ \text{✓ Preview Record} & & \text{No Previous Record} \end{tikzcd}$$

✓ Preview Record

X

No Previous Record

✓

IP Address

questions

2000/

politely

Contact No → $\boxed{9654617700}$

Priyanka maam

3

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| arr → | 3 | 0 | 5 | 1 | 10 | 6 | 2 | 10 | 4 | 0 | 1 | 1 | 9 | 2 |
| index → | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

left = 3
right = 8

prev - count = 1
count = ~~1~~ 2

0 → 0 x
3 0 → 3 ✓

j i

i = 0
j = 0

prev_count = 0
count = 0

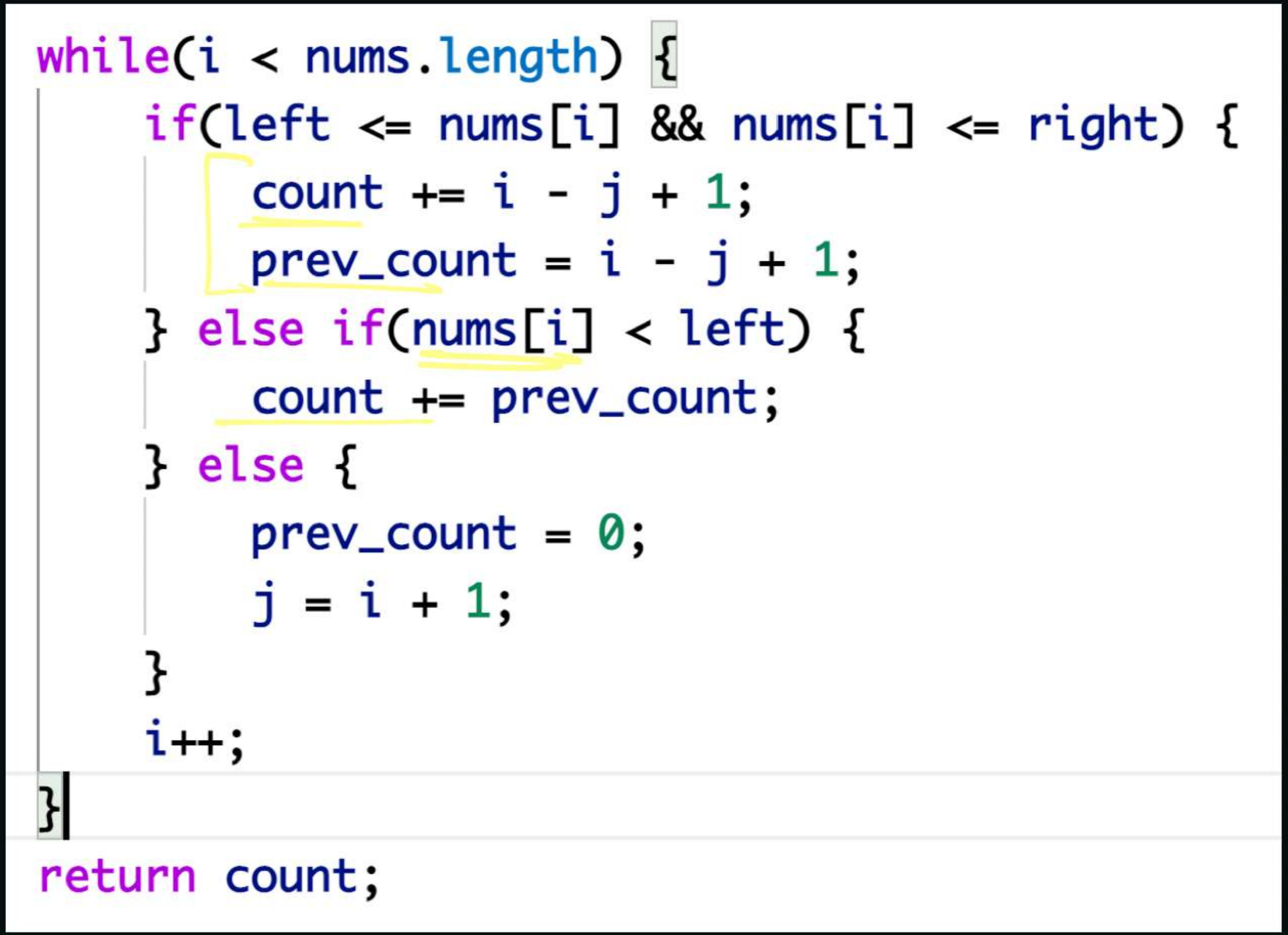
prev_count = ~~1~~ ~~3~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ 0
count = ~~1~~ ~~2~~ ~~4~~ ~~7~~ 8 ~~9~~ ~~10~~ ~~11~~ ~~12~~ ~~13~~ 14 = 14 Ans

✓3 → 3
✓3 0 → 3
✓5 → 5
✓0 5 → 5
✓3 0 5 → 5
✓5 1 → 5
✓0 5 1 → 5
✓3 0 5 1 → 5

✓6 → 6
✓6 2 → 6
✓4 → 4
✓4 0 → 4
✓4 0 1 → 4
✓4 0 1 1 → 4

All max are in Range

May be mistake in dry Run



```
while(i < nums.length) {
    if(left <= nums[i] && nums[i] <= right) {
        count += i - j + 1;
        prev_count = i - j + 1;
    } else if(nums[i] < left) {
        count += prev_count;
    } else {
        prev_count = 0;
        j = i + 1;
    }
    i++;
}
return count;
```

# Wiggle Sort 1

Friday, 17 September 2021 7:42 PM

arrange element of array in following Equality →

✓ $arr[0] <= arr[1] >= arr[2] <= arr[3] >= arr[4]$ - - - - -

In general → # odd Index Elements are greater then or equal to from next and prev element.

# even index is obvious then.

Index → 0 1 2 3 4

Element → b ~~a~~ ≤ ~~b~~ a ≥ c d e

i i

given

$a > b$

$b > c$

$c < b < a$

$c < b$ ⟹ $c < a$

```
if( i%2 ==1 ) {
    // odd Index
    if( arr[i] < arr[i+1]){
        swap(arr, i, i+1);
    }
}
else {
    // Even index
    if( arr[i] > arr [i+1]){
        swap(arr, i, i+1);
    }
}
```

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| arr → | 3 | 12 | 1 | 6 | 4 | 5 |

↑ ↑

arr[0] ≤ arr[1] ≥ arr[2] ≤ arr[3]
-------

$$\underbrace{3 \le 12 \ge 1 \le 6 \ge 4 \le 5}$$

Equality is maintained.

zig-zag pattern.

If Equality will be ↓

arr[0] ≥ arr[1] ≤ arr[2] ≥ arr[3] ≤ arr[4]---

change in testing Area

```
if(i%2==1) {
  // odd Index
  if( arr[i] < arr[i+1]){
     swap(arr, i, i+1);
  }
}
else {
  // Even Index
  if( arr[i] > arr [i+1]){
     Swap(arr, i, i+1);
  }
}
```

# Wiggle Sort 2

Friday, 17 September 2021 7:42 PM

allowed space → O(n) ⟶ ]||≡
allowed time → O(nlogn) ⟶ ]||≡

arr → arr[0] < arr[1] > arr[2] < arr[3] - - - - - -

NOTE: input is always valid ⟶

arr →

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 2 | 1 | 4 | 1 | 3 | 9 | 1 | 8 | 6 | 1 |

Equal case in centre

↓ → Duplicate

① Sort ⟶

1 1 1 1 1 2 3 4 6 7 8 9

min

mid : < middle

change in

② Actual array.

1 1 1 2 2 2 2 4 6 7 8 9

middle

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 < | 9 > | 1 < | 8 > | 1 < | 7 > | 1 < | 6 > | 1 < | 4 > | 1 < | 3. |

i i i

Step

middle ~~m~~, max

min middle

2 < 9 > 2 < 8 > 2 < 7 > 2 < 6 >, 1 < 4 > 1 < 2 > 1

Steps:

① make a duplicate array.

② sort duplicate array

③ Make iteration from $i = 1$ and increment with 2 steps until end, simultaneously $j$ is moving in duplicate array from end to start by 1

③ if $i$ reached at end, then $i$ again start with '0' and make 2 steps ahead.

# Range Addition

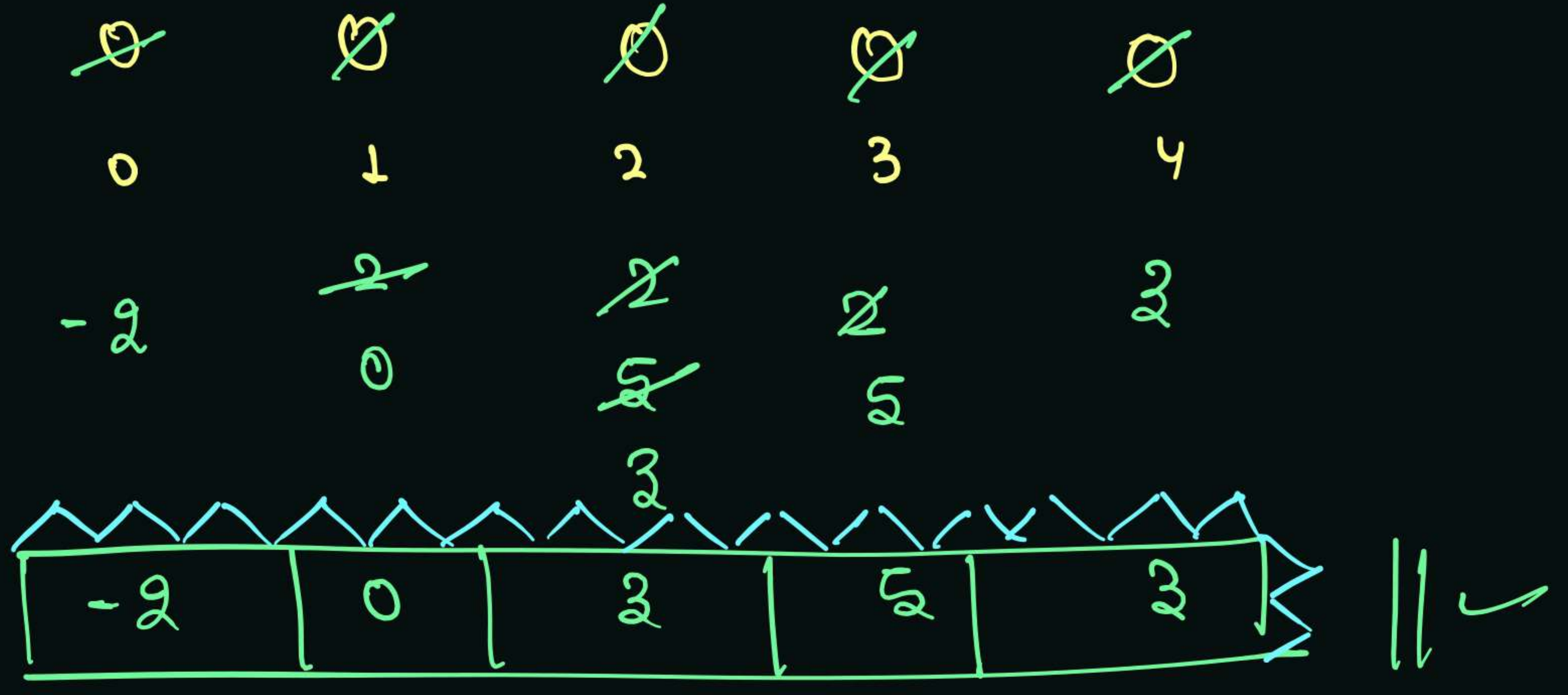
Friday, 17 September 2021 8:38 PM

length = 5,
updates =
[
[1, 3, 2],
[2, 4, 3],
[0, 2, -2]
]

operations

max. Range of query = $\boxed{n}$

q - query



operation → Starting Index Ending Index Increment

Brute force time - complexity = $O(qn)$

optimised Approach - allowed time complexity = $O(q+n)$ ||

Brute force - ① Iterate on 'q' queries and solve every query one by one.
② time complexity of brute force → O(qn)

Steps:

① n - size array.

② make Increment of Starting Index and Decrement on ending index +1, (ending index == n-1) → skip decr

③ Make prefix sum array.

optimised Approach → n=12

| query → | Starting Index | Ending Index | Incr. Dec. |
|---|---|---|---|
| ✓ | 2 | 6 | $\boxed{4}$ |
| ✓ | 0 | 5 | 3 |
| ✓ | 1 | 4 | -2 |
| ✓ | 6 | 9 | 1 |

q to mark impact

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ~~0~~ | ~~0~~ | ~~0~~ | 0 | 0 | ~~0~~ | ~~0~~ | ~~0~~ | 0 | 0 | ~~0~~ | 0 |
| ~~3~~ | ~~-2~~ | ~~4~~ |  |  | +2 | ~~-2~~ | -4 |  |  | -1 |  |
|  |  |  |  |  |  | -2 |  |  |  |  |  |
| 3 | 1 | 5 | 5 | 5 | 7 | 5 | 1 | 1 | 1 | 0 | 0 |

n → to make / visible impact

Result

[0 3 -1]

2-6 → 4 →

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
|  | 1 | 0 | 4 | 0 | -1 | 0 | 0 | -4 |
| prefix → | 1 | 1 | 5 | 5 | 4 | 4 | 4 | 0 |

4

| st | end | val |
|---|---|---|
| 1 - | 5 = | $\boxed{3}$ |
| 0 → | 3 → | 1 |
| 1 - | 7 → | -2 |

prefix sum

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ~~0~~ | ~~0~~ | 0 | 0 | ~~0~~ | 0 | ~~0~~ | 0 |
| 1 | ~~2~~ 2 1 | | | -1 | | -3 | |

$+2$

prefix

| 1 | 2 | 2 | 2 | 1 | 1 | -2 | -2 |
|---|---|---|---|---|---|---|---|

Array and String ↳ Topic of cramming.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | 3 | 3 | 3 | 3 | 3 | | |
| 1 | 4 | 4 | 4 | | | | |
| 1 | 2 | 2 | 2 | 1 | 1 | -2 | -2 |

# Product of array except itself

Friday, 17 September 2021 9:35 PM

arr → 1 2 3 4 5

result → | 120 | 60 | 40 | 30 | 24 |

$\frac{120}{1}$ $\frac{120}{2}$ $\frac{120}{3}$ $\frac{120}{4}$ $\frac{120}{5}$

120 60 40 30 24 || ✓

X

Exception

1 2 3 0 5

$\frac{0}{1}$ $\frac{0}{2}$ $\frac{0}{3}$ $\left[\frac{0}{0}\right.$ $\frac{0}{5}$

0 0 0 $\boxed{30}$ 0

Arithmatic Exception

① Approach - 0

product = 5×4×3×2×1
= 120

product = 0
$1\times2\times3\times0\times5$

→ Divide by 0 Error

→ Division operator is not allowed.

|  |  |  |  |  |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 120 | 120 | 60 | 20 | 5 | 1 |

right product array

left product array →

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 6 | 24 | 120 |

result

|  |  |  |  |  |
|---|---|---|---|---|
| 120 | 60 | 40 | 30 | 24 |

product Except itself

steps

1. Right product array
2. left will manage at runnig time

# Pascal's Triangle 2

Friday, 17 September 2021 9:51 PM

$n \equiv$ Row

$r \equiv$ Column

$^{n}C_{r} * F = {}^{n}C_{r+1}$

$$\frac{\cancel{n!}}{(n-r)!\,r!} * F = \frac{\cancel{n!}}{(n-r-1)!\,(r+1)!}$$

$$F = \frac{(\overset{p}{n-r})!\; r!}{(n-r-1)!\;(r+1)!}$$

$[n! = n * (n-1)!]$

$$F = \frac{(n-r) * \cancel{(n-r-1)!} * \cancel{r!}}{\cancel{(n-r-1)!}\;(r+1) * \cancel{r!}}$$

$^{row}C_0 \quad ^{row}C_1 \quad ^{row}C_2 \ldots \quad ^{row}C_{col}$

1

P!

$$\boxed{F = \frac{n-r}{r+1}}$$

$\Rightarrow$

$$\boxed{^{n}C_{r} * \frac{n-r}{r+1} = {}^{n}C_{r+1}}$$

## Properties of Combination

# $^{n}C_0 = 1$ $\quad 0! = 1$

# $^{n}C_n = 1$

# $^{n}C_r = {}^{n}C_{n-r}$

# $^{n}C_r = \frac{n!}{(n-r)!\,r!}$

# $^{n}C_r * \underset{\sim}{F} = {}^{n}C_{r+1}$