Today's Agenda :-

Will start at 9:05 PM

1) Imp Announcement
2) Count of Pairs "ag"
3) No of Leaders in an array
4) Intro to Subarrays
5) Smallest Subarray containing min & max of array

Q1) Given a String S of lowercase characters. Return the count of pairs (i, j) such that $i < j$ & $S[i] = 'a'$ & $S[j] = 'g'$.

Eg Str: "abegag"

```
  0 1 2 3 4 5
" a b e g a g "
```

(0, 5)
(4, 5)
(0, 3)

Eg Str: "acgdgag"

```
  0 1 2 3 4 5 6
" a c g d g a g "
```

i	j
0	2
0	4
0	6
5	6

ans = 4

Eg Str: "bcaggaag"

```
  0 1 2 3 4 5 6 7
" b c a g g a a g "
```

i	j
2	3
2	4
2	7
5	7
6	7

ans = 5

Brute force :-

```
int cut = 0
for(int i = 0; i < N; i++) {          // i < N-1
    if (s[i] == 'a') {
        for(int j = i+1; j<N; j++) {
            if (s[j] == 'g') {
                cut ++
            }
        }
    }
}
```

str: g g a g a a (indices 0 1 2 3 4 5; i)

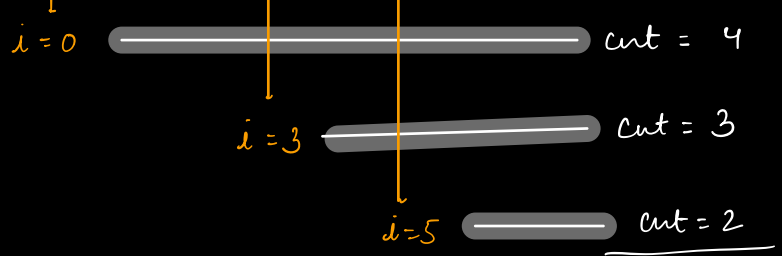
s[i] → 0, N-1

TC : O(N²)
SC : O(1)

Observation :-

For every 'a', we are counting 'g' on right side

Eg Str: a d g a g a g f g (indices 0 1 2 3 4 5 6 7 8)



ans = 9

Right to left

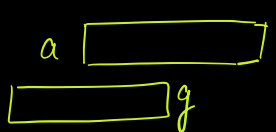
Maintaining count of g

Eg

a	d	g	a	g	a	g	f	g
ans = ans + cnt_g ans = 9		cnt_g = 4	ans = ans + cnt_g ans = 5	cnt_g = 3	ans = ans + cnt_g ans = 2	cnt_g = 2		ans = 0 cnt_g = 0 cnt_g = cnt_g + 1 cnt_g = 1

```
int ans = 0, cnt_g = 0
for(int i = N-1; i >= 0; i--) {
    if (str[i] == 'g') {
        cnt_g++
    }
    else if (str[i] == 'a') {
        ans = ans + cnt_g
    }
}
return ans
```

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



TODO: Try going from left to right

Hint : You have to maintain cnt_a

⟶

Q2) Given an array of size N. Count the no of leaders.

Leader : An element strictly greater than all elements on the right side.

Eg ar[] : {2, 5, 3, 4, 17, 16} } ans = 2 ✓

Eg ar[] : {2, 5, 4, 3, 2, 1} } ans = 5

Eg ar[] : {10 8 8} } ans = 2 ✓

BF : For every element at i^{th} index, if all elements on right side are less than ar[i], then i^{th} element will be a leader.

```
int leaders = 1
for (int i = 0; { i <= N-2 OR i < N-1 }; i++) {
    bool flag = true
    for (int j = i+1; j < N; j++) {
        if (ar[j] > ar[i]) {
            flag = false, break;
        }
    }
    if (flag == true) { leaders ++ }
}
return leaders
```

TC : O(N²)
SC : O(1)

Optimisation :-

1) For every element, we are considering all elements on right

2) Scan all elements from R → L & maintain track of max till now

```
          i
     0  1  2  3  4  5  6  7
ar[]: { 7  13 2  9  12 6  8  4 }
```

last_max = 4
cnt = 1

i	ar[i]	last_max	count of leaders
6	8	~~4~~ 8	2
5	6	8	2
4	12	~~8~~ 12	3
3	9	12	3
2	2	12	3
1	13	~~12~~ 13	4
0	7	13	4

cnt = 4

```
int leaders = 1
int last_max = ar[N-1]
for(int i = N-2; i >= 0; i--) {
    if (ar[i] > last_max) {
        leaders++
        last_max = ar[i]
    }
}
return leaders
```

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

Subarray :- Commonly defined as part / section of an array which consist of contiguous elements of array.

Eg :- ar[ ] : { 1, 5, 2, 3, 6, 8, 4 }

0 1 2 3 4 5 6

[5, 2, 3, 6] : [1 4]

s e

Subarray [s, e] : $e - s + 1$ → length of subarray

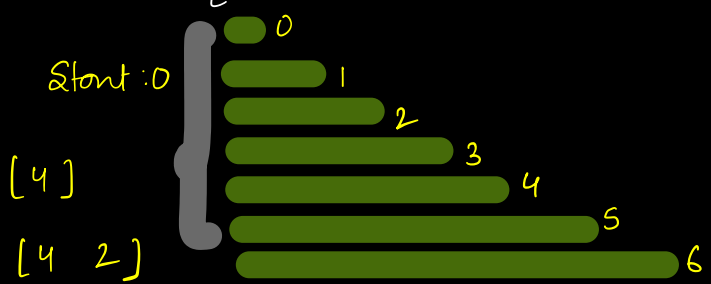
↳ $s <= e$

- [1, 1]

- A single element is a subarray : 1 2, 3, 4 . . .
- An entire array is also a subarray : N

Eg ar[ ] : { 4, 2, 10, 3, 12, -2, 15 }

0 1 2 3 4 5 6



[4]

[4 2]

[4, 2, 10]

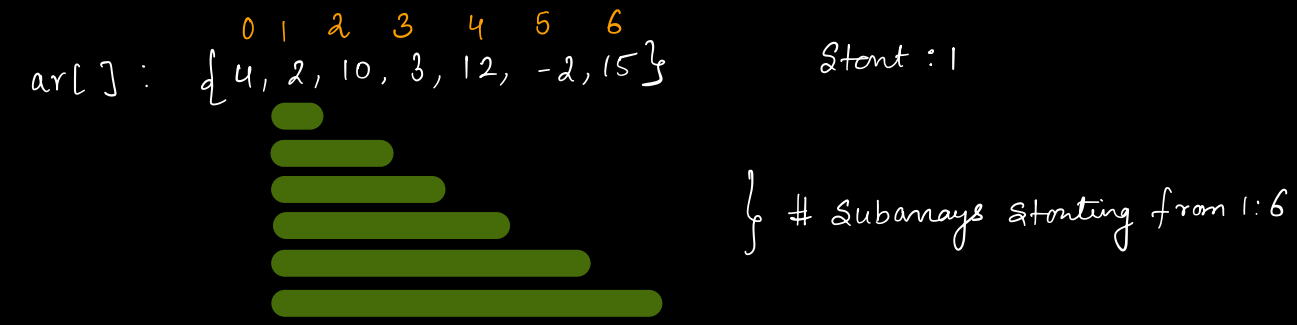
[4, 2 10 3]

[4 2 10 3 12]

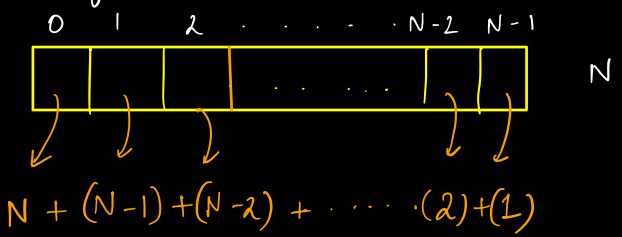
[4 2 10 3 12 -2]

[4 2 10 3 12 -2 15]

subarays starting from 0 : 7



Total No of subarrays for a given array of size N :-



Sum of N Natural Nos

$$\#\text{ of Subarrays} = \frac{N(N+1)}{2}$$

Break for 7 Min Relax, Have some water !

Contest : Online coding assessment / challenge

↳ 11th Aug (friday) : 1st Contest }

↳ 9:PM – 10:30PM (90 min) → 3 coding Ques

10:30 PM – Contest Discussion ✓

Syllabus = : Arrays + TC }

1) Attend session
2) Solve assignments

Q3) Given an array of size N. Return the length of the smallest subarray which contains both maximum & minimum element of the array.

Eg arr[] : { 2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1 }
(indices 0 1 2 3 4 5 6 7 8 9 10)

min_ele = 1
max_ele = 6

s e
[8, 10] : 3 ans

Eg arr[] : { 1, 2, 3, 1, 3, 4, 6, 4, 6, 3 }
(indices 0 1 2 3 4 5 6 7 8 9)

1 & 6

[0 8] : 9

min_ele = 1
max_ele = 6

s e
[3 6] : 6 - 3 + 1 : 4 = ans

Eg : arr[] : { 8 8 8 8 8 }
(indices 0 1 2 3 4)

min_ele = 8
max_ele = 8

[0 0] ✓ len = 1 ✓
[1 1] ✓
[2 2] ✓

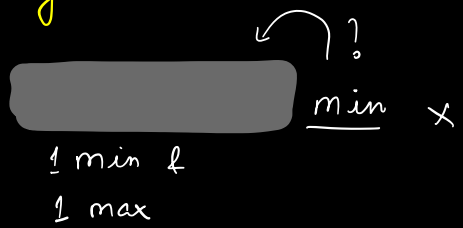
BF: for all subarrays that exist, find the smallest len which contains both min & max.

↳ Next class : Subarrays

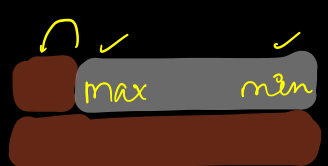
↳ TC : $O(N^3)$
SC : $O(1)$

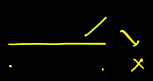
Observations :- Ans subarray

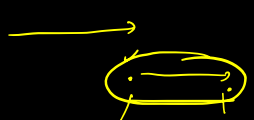
1) final ans subarray should contain 1 min & 1 max



2) Max & Min should be only on corners



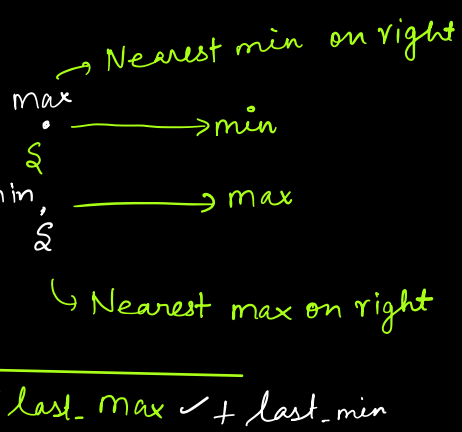




↳ ans subarray

3) Case 1 [max min]

Case 2 [min max]



ans = N

min_ele = 1
max_ele = 6
last_min = 5
last_max = 8

Eg ar[]: { 2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1 }

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

last_max = 2
[2, 5] : 4
ans = 3

last_min = 5
[5, 8] : 4
ans = 3

last_max = 8
[8, 10] : 3
ans = 3

last_min = 10

ans = 3

```
int min_ele
int max_ele
int last_min_i = -1
int last_max_i = -1
int ans = N
if (min_ele == max_ele) return 1
// iterate & find min_ele & max_ele of array
for (int i = N-1; i >= 0; i--) {
    if (ar[i] == min_ele) {
        last_min_i = i
        // [ i  last_max_i ] : possible candidate
        if (last_max_i != -1) {
            int len = last_max_i - i + 1
            if (len < ans) { ans = len }
        }
    }
    ...
```

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

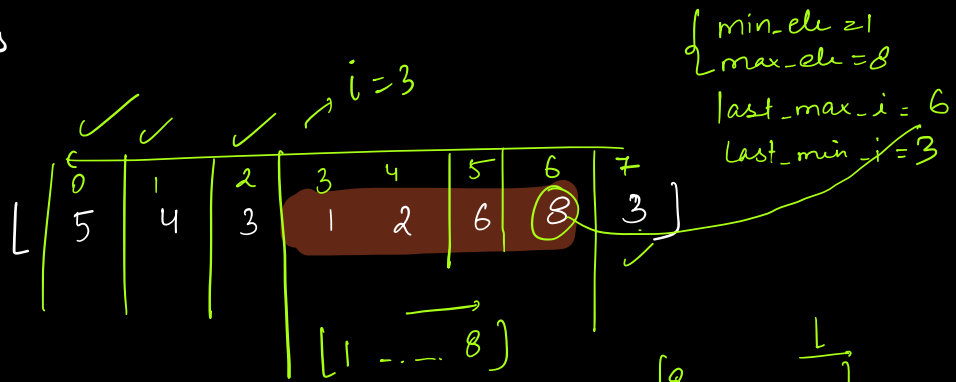
[s e]
e - s + 1

i → last_max_i
min_ele →

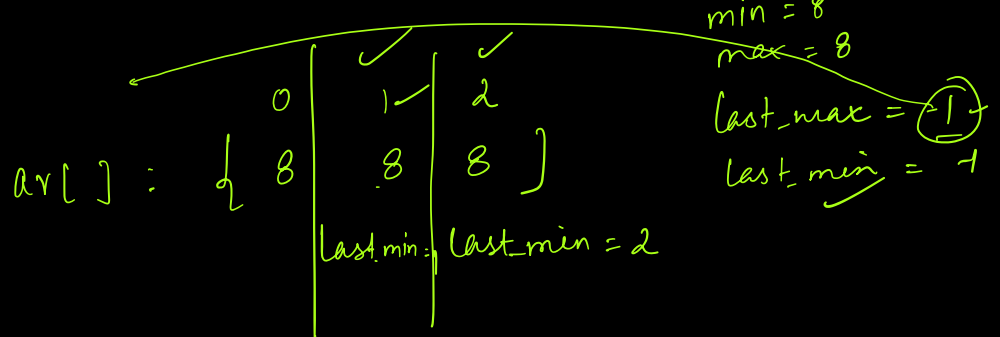
```
if (arr[i] == max_ele) {
    last_max_i = i
    // [ i ... last_min_i ] : possible candidate
    if (last_min_i != -1) {
        int len = last_min_i - i + 1
        if (len < ans) { ans = len }
    }
}
}
return ans
```

s e
[ith → last_min
max_ele

Eg :-



Eg



Next class :- **Subarrays**

Agenda :- 1) Printing all Subarrays

2) Sum of all Subarrays in different ways → O(N) ✓

3) Contribution Technique } Very Important ✓

The concepts learned will be used throughout the course, So requesting all to not miss the session !

Thank You