Q1 Count pairs "a g".

Given a character array, calculate the no of pairs i,j such that.

i<j && s[i] = 'a' && s[j] = 'g'.

All characters are lower case.

Ex1 : arr[] = b a a g d c a g (indices 0 1 2 3 4 5 6 7)

Pairs : [ <1,3>, <2,3>, <1,7>, <2,7>, <6,7> ]

ans = 5

Brute force

```
int c = 0;
for (int i=0 ; i<n ; i++) {
    if (s[i] == 'a') {
        for (int j = i+1 ; j<n ; j++) {
            if (s[j] == 'g')
                c++;
        }
    }
}
return c;
```

Tc : O(n^2)

Sc : O(1)

Ex1 : arr[] =

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| b | a | a | g | d | c | a | g |

Prefix Sum Array for a : 0 1 2 2 2 2 3 3

no of a's encountered till ith index

ans = 0 → 2 → 5

Tc: O(n)

Sc: O(n)

1) Original array cannot be modified to store prefix sum as it is a char array.

int ans = 0 , int cnt_a = 0

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| b | a | a | g | d | c | a | g |

| index | S[i] | |
|---|---|---|
| 0 | b | ans = 0<br>cnt_a = 0 |
| 1 | a | ans = 0<br>cnt_a = 1 |
| 2 | a | ans = 0<br>cnt_a = 2 |
| 3 | g | ans += cnt_a \| ans = 2<br>cnt_a = 2 |
| 4 | d | ⟶ |
| 5 | c | ⟶ |
| 6 | a | ans = 2<br>cnt_a = 3 |
| 7 | g | ans = 5<br>cnt_a = 3 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| b | a | a | g | d | c | a | g |

1) Count all occurences of g

cnt-g = 2

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| g | a | a | g | d | c | a | g |

cnt-g = 2

## Pseudo Code

```
int ans = 0
int cnt_a = 0
for (int i=0 ; i<n ; i++ ) {
        if (s[i] == 'a')
                cnt_a++;
        else if (s[i] == 'g')
                ans = ans + cnt_a
}
return ans;
```

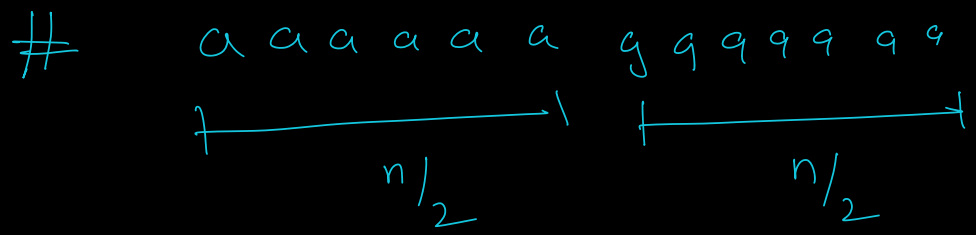
TC: O(n)

SC: O(1)

```
int ans = 0
int list<int> index_a;
for (int i=0 ; i<n ; i++ ) {
        if (s[i] == 'a')
                index_a.add(i);
        else if (s[i] == 'g')
                print_pairs(index_a, i)
}
return ans;
```

SC: O(n)

TC: O(n²)


#
a a a a a a
g g g g g g
n/2
n/2

Q2 Leaders in a Array

Given an array, you have to find count of all leaders in arr[].

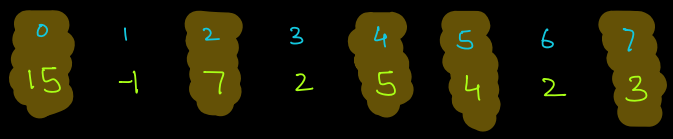
All element is a leader, if it is strictly greater than all elements on it's right side.

Note: arr[n-1] is always considered as Leader.

Ex1 : arr[] = 

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 15 | -1 | 7 | 2 | 5 | 4 | 2 | 3 |

ans = 5




int curr_max = 3
int ans = 1

| index | arr[i] | |
|---|---|---|
| 6 | 2 | curr_max = 3<br>ans = 1 |
| 5 | 4 | curr_max = 4<br>ans = 2 |
| 4 | 5 | curr_max = 5<br>ans = 3 |
| 3 | 2 | curr_max = 5<br>ans = 3 |
| 2 | 7 | curr_max = 7<br>ans = 4. |
| 1 | -1 | curr_max = 7<br>ans = 4. |
| 0 | 15 | curr_max = 15<br>ans = 5 |
| | | |

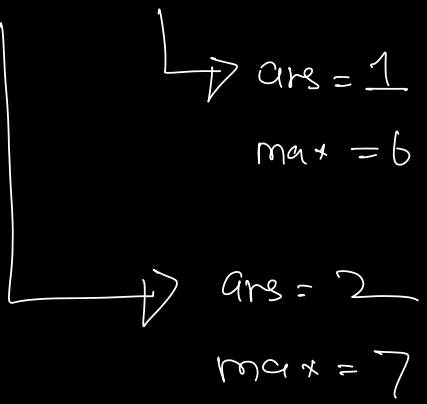
## Pseudo Code

```
int curr_max = arr[n-1];
int ans => 1

for (int i = n-2 ; i >= 0 ; i--) {
    if (arr[i] > curr_max) {
        ans ++;
        curr_max = arr[i]
    }
}

return ans;
```

TC : O(n)

SC : O(1)

# 2 4 4 7 7 6 6



# Subarray Basics

→ Continuos part of an array.

1) Single element ✓

2) Entire array ✓

3) Empty array. ✗

# arr[] =

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

| indices | Subarray |
|---|---|
| [1, 2, 3, 4] | ✓ |
| [1, 2, 4] | ✗ |
| [4, 5] | ✓ |

Length of a subarray $(i, j) = (j - i + 1)$

No of Subarrays in a array of length n.

| Start | End. | |
|---|---|---|
| 0 | [0, n-1] | ⇒ n |
| 1 | [1, n-1] | ⇒ n-1 |
| 2 | [2, n-1] | ⇒ n-2 |
| ⋮ | ⋮ | ⋮ |
| n-1 | [n-1, n-1] | ⇒ 1 |

Total Subarrays ⇒ $\frac{n(n+1)}{2}$

# Q3 Closest Min Max

Given an array find the length of smallest subarray which contains both Min & Max of array.

Ex1 arr[] :

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

Potential Subarrays :
1) [3, 7]
2) [0, 6]
3) [0, 8]
4) [3, 8]
5) [3, 6]

min = 1
max = 6

ans = 4

Ex2 arr[] :

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 8 | 8 | 8 | 8 |

ans = 1

Observations

1) Min & Max should be at ends of subarray.
2) Min == Max ans = 1
3) There would be only 1 min & 1 max.


# Subarray ⇒ [min, max]

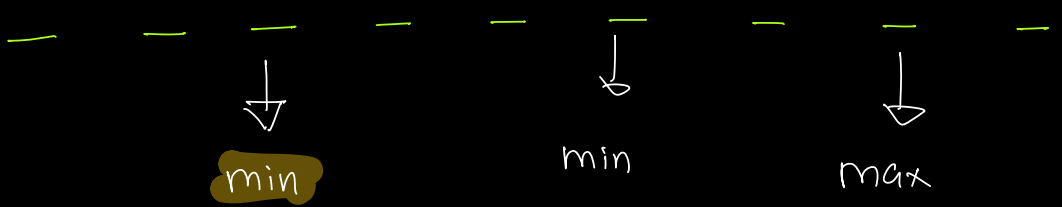
[max, min]

## Approach 1

1) Find min & max in 1 loop.
2) Loop again & store indices of min & max.

indices_min = [ ]

indices_max = [ ]




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

| index | arr[i] | |
|---|---|---|
| 9 | 3 | —— |
| 8 | 6 | max_index = 8<br>min_index = -1 |
| 7 | 4 | —— |
| 6 | 6 | max_index = 6<br>min_index = -1 |
| 5 | 4 | —— |
| 4 | 3 | —— |
| 3 | 1 | max_index = 6<br>min_index ⇒ 3 |
| 2 | 3 | —— |
| 1 | 2 | —— |
| 0 | 1 | max_index = 6<br>min_index = 0 |

min_index = -1

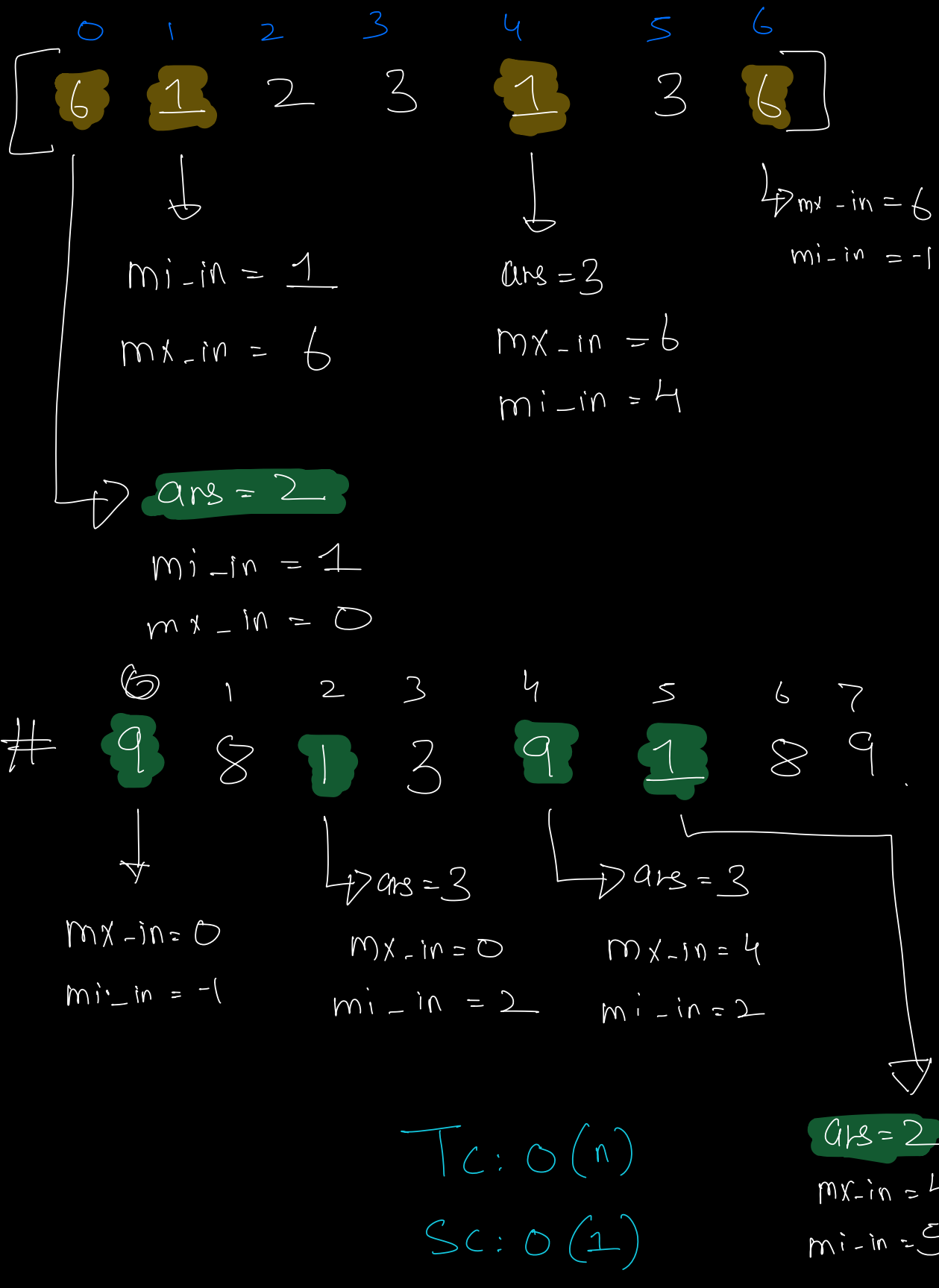
max_index = -1

ans ⇒ 0

max = 6

min = 1

ans ⇒ 4


0 1 2 3 4 5 6
6 1 2 3 1 3 6
mi_in = 1
mx_in = 6
ans = 3
mx_in = 6
mi_in = 4
mx_in = 6
mi_in = -1
ans = 2
mi_in = 1
mx_in = 0
0 1 2 3 4 5 6 7
# 9 8 1 3 9 1 8 9
mx_in = 0
mi_in = -1
ans = 3
mx_in = 0
mi_in = 2
ans = 3
mx_in = 4
mi_in = 2
ans = 2
mx_in = 4
mi_in = 5
Tc: O(n)
Sc: O(1)

Doubts

1) Can Questions be unlocked when Doubt session starts.

2) Pseudo Code

```
if (arr[i] == min && max_ind != -1)
    min_in => i;
    len => max_ind - i + 1
else if (arr[i] == max && min_ind != -1)
    max_in = i;
    len => min_ind - i + 1
```

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