Q1 Count pairs "ag"

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.

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

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

ans = 5

Pseudo Code [Brute force]

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

Tc: $O(n^2)$

Sc: $O(1)$

Ex1 arr[] = b a a g d c a g

(0 1 2 3 4 5 6 7)

cnt = 0 , cnt_g = 0

b	a	a	g	d	c	a	g
no Charge	cnt_g = 2 cnt = 5	cnt_g = 2 cnt = 3	cnt_g = 2 cnt = 1	no Charge	no Charge	cnt_g = 1 cnt = 1	cnt_g = 1 cnt = 0

ans = 5

Ex1 arr[] = a a g g a g g g

(0 1 2 3 4 5 6 7)

cnt ⇒ 13
cnt_g ⇒ 5

cnt ⇒ 8
cnt_g ⇒ 5

cnt ⇒ 3
cnt_g ⇒ 3

Pseudo Code

```
int cnt = 0, cnt_g = 0

for (int i = n-1 ; i ≥ 0 ; i--) {

    if (arr[i] == 'g')
        cnt_g++;
    else if (arr[i] == 'a')
        cnt += cnt_g;

}

return cnt;
```

TC: O(n)

SC: O(1)

Q2 Leaders in a Array.

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

An element is a leader, if it is strictly greator 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

Brute force : Tc: $O(n^2)$ [run 2 loops]

max = arr[n-1] 3

cnt = 1

	0	1	2	3	4	5	6	7
arr[] =	15	-1	7	2	5	4	2	3
	cnt=5	✗	cnt=4	cnt=3	cnt=3	cnt=2	cnt=1	
	max=15		max=7	max=5	max=5	max=4	max=3	

Pseudo Code

```
max_val = arr[n-1];
cnt = 1;

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

return cnt;
```

Subarray Basics

→ Continuous part of an array.

1) Single Element ✓

2) Full array. ✓

3) Empty array ✗

Ex1 arr[] =

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

indices	Subarray.	
[1, 2, 3, 4]	✓	= [1, 4]
[1, 2, 4]	✗	=
[4, 5]	✓	= [4, 5]

length of a subarray [i, j] = $j - i + 1$

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

(i) [3, 6] ⇒ 4
(ii) [0, 6] = 7
(iii) [0, 8] = 9
(iv) [3, 8] ⇒ 6

min = 1
max = 6

Ex2 arr[] =

0	1	2	3
8	8	8	8

ans = 1

No of Subarrays in a arr of length n.

$$arr[] = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [a_1 & a_2 & a_3 & a_4 & a_5] \end{matrix}$$

Start	no of ending points	
0	5	[0, 1, 2, 3, 4]
1	4	[1, 2, 3, 4]
2	3	[2, 3, 4]
3	2	[3, 4]
4	1	[4]

Start	ending points	count
0	[0, n-1]	n
1	[1, n-1]	n-1
2	[2, n-1]	n-2
⋮		
n-1	[n-1, n-1]	1

$$\frac{n(n+1)}{2}$$

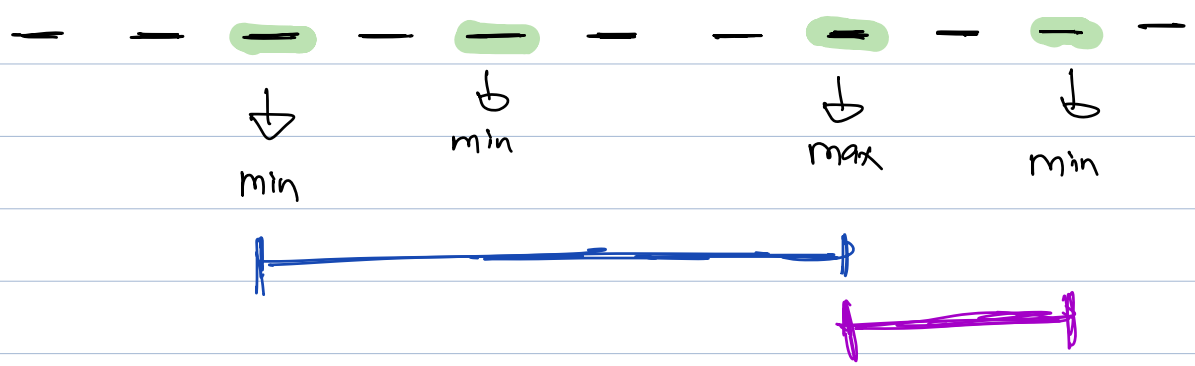
Observation

	0	1	2	3	4	5	6	7	8	9
arr[] =	1	2	3	1	3	4	6	4	6	3

1) Min & Max would be endpoints of the subarray.

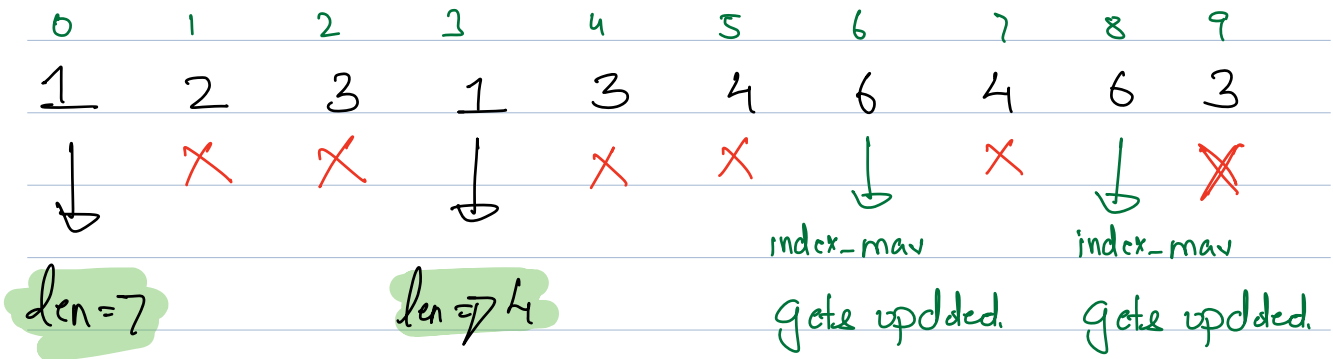
2) Only 1 min & 1 max in subarray.

3) CASE 1 : [min, max]
   CASE 2 : [max, min]



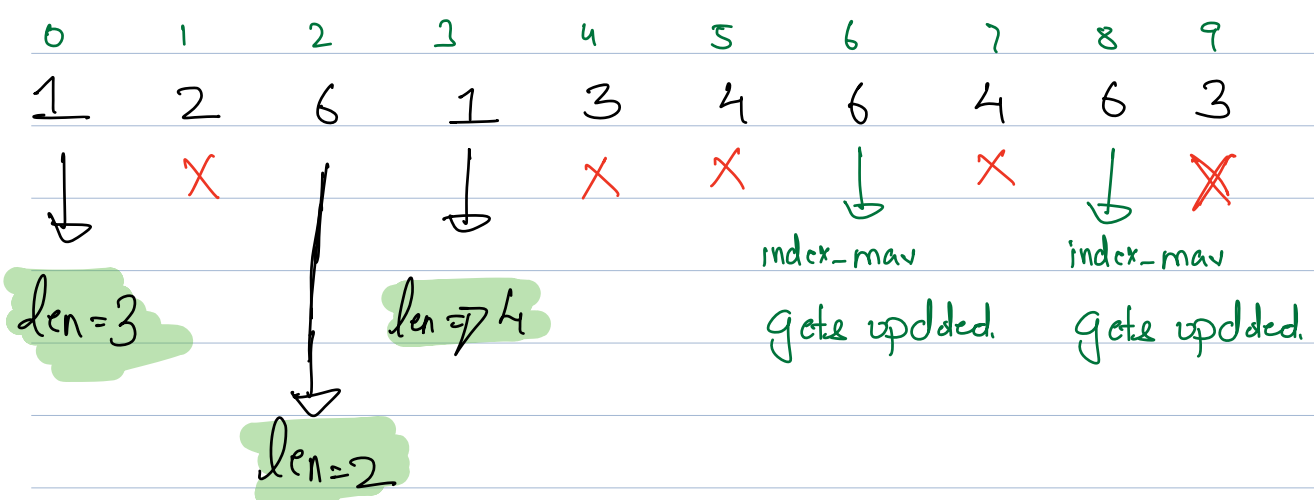
index_max = 6 min = 1 ans = 4

index_min = 3 max = 6



index_max = 2 min = 1 ans = 2

index_min = 3 max = 6



index_max = 8 min = 1 ans = 4

index_min = 3 max = 6

0	1	2	3	4	5	6	7	8	9
1	2	3	1	3	4	6	4	6	3
↓	X	X	↓	X	X	↓	X	↓	X

update index_min

update index_min

len=4 update max

len=6 update max

0	1	2	3	4	5	6	7	8
1	2	3	6	2	3	1	1	6

min_index = 7

max_index = 3

ans = 2

Predefined function.

int a, int b;

max (a, b);

min (a, b);

arr [];

sort (arr);

int a = $[-2 \times 10^9, 2 \times 10^9]$

long a = $[-2 \times 10^{18}, 2 \times 10^8]$

a	n	an	a+n
1	1	1	2
2	2	4	4