Ques) Count Pairs "ag"

Given a char[], calculate no. of pairs i,j s.t, i<j, s[i]=='a' && s[j]=='g'.

Note:- All characters are lower case.

e.g.)

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

Pairs:- <1,3>, <2,3>, <1,7>, <2,7>, <6,7> => 5 pairs.

e.g 2)

0	1	2	3	4	5	6	7
a	c	a	g	g	a	a	g

Pairs :- <2,3>, <2,4>, <2,7>, <5,7>, <6,7> => 5 pairs

e.g.3)

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

Pairs :- <0,2>, <0,4>, <0,6>, <5,6>

=> 4 pairs

idea 1) a) check all pairs (i, j)

T.C → O(N^2)

S.C → O(1)

```
cnt = 0;
for (i=0; i<N; i++) {
    for (j=i+1; j<N; j++) {
        if (s[i] == 'a' && s[j] == 'g')
            cnt++
    }
}
```

idea 2) a) check all pairs (i, j)

T.C → O(N^2)

S.C → O(1)

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

0	1	2	3	4	5	6	7	8
a	d	g	a	g	a	g	f	g

cnt = 4

cnt++ = 3

cnt++ = 2

9

C → how many g's are to the right.

Ans = 0 C = 0

0	1	2	3	4	5	6	7	8
a	d	g	a	g	a	g	f	g
Ans = Ans + C = 5 + 4 => 9		C = C+1 C = 4	Ans = 3 Ans = 5	C = C+1 C = 3	Ans = 2	C = C+1 C = 2		C = C+1 C = 1

C = 0, ans = 0

```
for (i = n-1; i >= 0; i--) {
    if (s[i] == 'g') {
        c++
    }
    if (s[i] == 'a') {
        ans = ans + c
    }
}
return ans.
```

T.C → O(N)
S.C → O(1)

20) Leaders in an Array.

Given an Ar[n], you have to count leaders in arr[].

An ele is leader if it is strictly greater than max of elements in its right.

Ex1: 15, -1, 7, 2, 5, 4, 2, 3 (indices 0 1 2 3 4 5 6 7)

Ans = 5

Ex2: 10, 7, 9, 3, 2, 4 (indices 0 1 2 3 4 5) → 3

Ex3: arr[7] = 8, -2, 4, 7, 6, 5, 1 (indices 0 1 2 3 4 5 6)

Ans = 5

Ex4:- 10, 8, 8 (indices 0 1 2) → 2

:. Bf :- for every element, iterate to right and get max.

T.C → O(N²)
S.C → O(1)

```
(i=0; i<n; i++)
    // iterate & get max to right.
    for(j=i+1; j<n; j++) {
        ___
        |
        |
        3
3
```

ans = 5
max = 15

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

```
ans = 1
max = arr[n-1]
for (i=n-2; i>=0; i--) {
    if (arr[i] > max)
    {   ans++;
        max = arr[i]
    3
3
```

T.C → O(N)
S.C → O(1)

10 8 8 4

Ans = 3
max = 10

Subarray Basics

1) Continuous part of an array is called subarray.

   1) A single element is a subarray
   2) full array is also a ''
   3) [ ], is also a subarray.

Ex :-

0	1	2	3	4	5	6	7	8
-3	4	6	2	8	7	14	9	21

indices [2, 3, 4, 5]

Ex2- indices [3, 4, 6, 7, 8] ✗

→ [1, 2, 3] ✓

→ [5] ✓

ind - [2, 8] ⇒ 8 - 2 + 1 ⇒ 7

ind [s e]
↳ e - s + 1

10 - 18 pm → 10:26 pm

T.C Sorting → O(nlogn)

Assuming we know, writing min and max of an array.

Ques) Closest min max :-

Given n array elements, find the length of smallest subarray, which contains both min and max of array.

Ex 1)

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

min = 1

max = 6

Ans = 4

Ex 2)

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	1

Observations :- 1) we need only 1 min & 1 max.

Subarray min min max

max min max

2) min and max will always be in corners.

3) [min max] or [max min]

Ex) 8 8 8 → 1

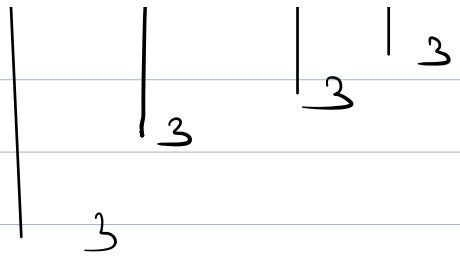
min = 1, max = 6 ans = 3

0	1	2	3	4	5	6	7	8	9	10	11	12
2	2	6	6	5	1	5	2	6	4	1	3	4

// iterate & get min_val & max_val.
if (min_val == max_val)
    return 1

T.C → O(n^2)
S.C → O(1)

```
ans = n
for (i=0; i<N; i++) {
    if (arr[i] == min_val) {
        for (j=i+1; j<N; j++) {
            if (arr[j] == max_val) {
                ans = min(ans, j-i+1)
                break;
            }
        }
    }
    if (arr[i] == max_val) {
        for (j=i+1; j<N; j++) {
            if (arr[j] == min_val) {
                ans = min(ans, j-i+1)
                break;
```



return ans;

// optimize

min = 1
max = 6

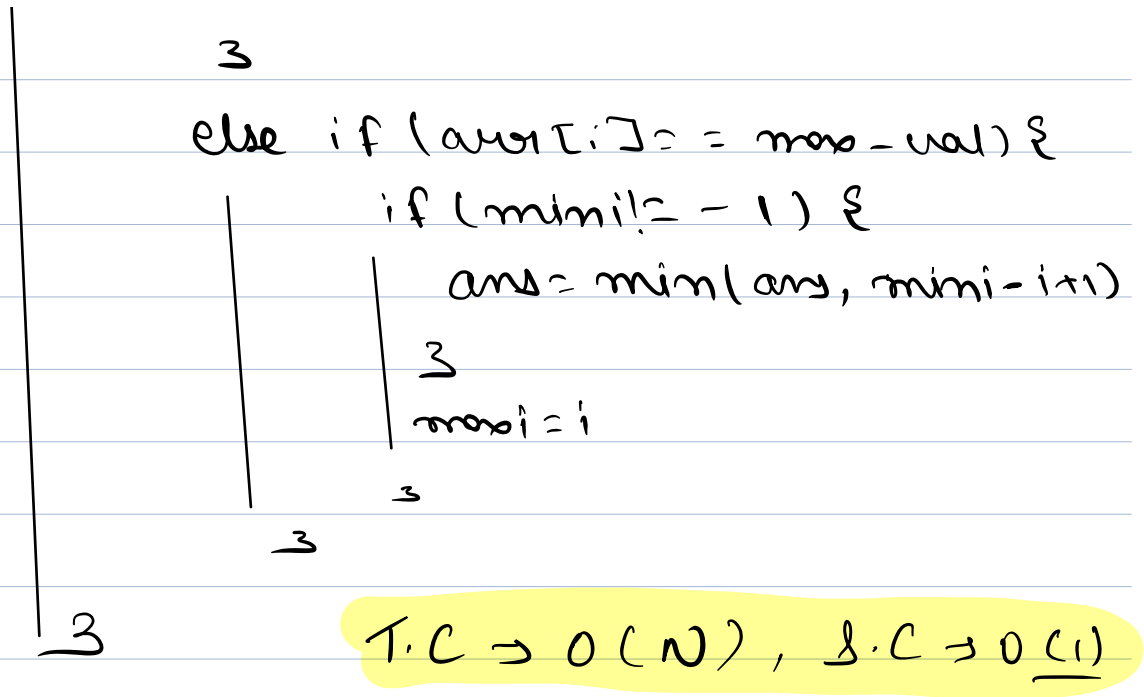
mini = ~~30~~
maxi = ~~8 31~~

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	6	4	6	5	1	5	2	6	4	4	2	1	5

Ans = ~~4 3~~ 2

1) Iterate & get min-val & max-val
2) if (min-val == maxval) return 1
   ans = 0, maxi = -1, mini = -1

```
for (i = n-1; i >= 0; i--) {
    if (arr[i] == min_val) {
        if (maxi != -1) {
            ans = min(ans, maxi - i + 1)
        }
        mini = i
```



T.C → O(N), S.C → O(1)