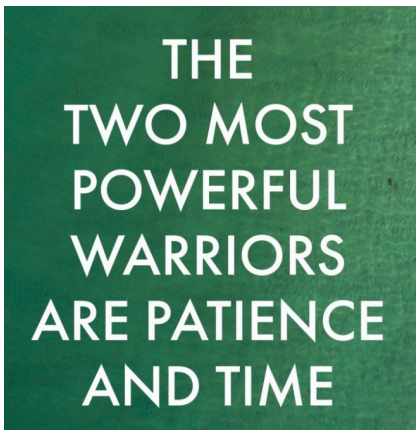
Today's Quote -



Questions on Carry forward.

Q) Count pairs "a g"

Given a char[], calculate no. of pairs i, j such that $i < j$ & $s[i] = 'a'$ and $s[j] = 'g'$. All characters are lower-case. {a should come before b}

Eg: { b a a g d c a g }
0 1 2 3 4 5 6 7

{1,3} {2,7}
{1,7} {6,7}
{2,3} {6,3}.

ans = 5

Q1) { b c a g g a a g }
0 1 2 3 4 5 6 7

{2,3} {5,7}
{2,4} {6,7}
{2,7}

ans = 5

Q2) { a c g d g a g }
0 1 2 3 4 5 6

{0,2} {5,6}
{0,4}
{0,6}

ans = 4

idea1: Consider all pairs.

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

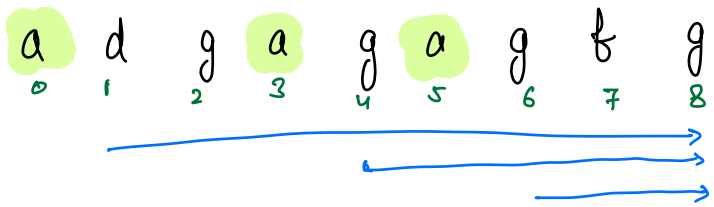
T.C → $O(N^2)$
S.C → $O(1)$

idea 2. : Consider only when s[i] == 'a'

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

T.C → O(N²)

Eg: a d g a g a g f g



idea :
Maintain count of g's from r.h.s.

a d g a g a g f g

0 1 2 3 4 5 6 7 8

c+=1 c=4 | c+=1 c=3 | c+=1 c=2 | c=1

ans += c ans = 9

ans += c ans = 5

ans += c ans = 2

ans = 9

Here, we are carrying forward count of g's from right hand side.

pseudo.code:

```
c = 0 , ans = 0

for ( i = N-1 ; i >= 0 ; i--) {
    if ( s[i] == 'g' ){
        c += 1 ;
    }
    else if ( s[i] == 'a' ){
        ans += c ;
    }
}

return ans
```

T.C → O(N)
S.C → O(1)

Alternative approach. → for every 'g', count the no. of a's on l.h.s.

todo: traverse the array from left to right & carry forward the count of a's.

Q2.) **Leaders in An Array**

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

An element is leader, if **it is strictly greater than max on right side.**

Note. → arr[N-1] is always considered as leader.
(last element)

Eg: { 15 (0), -1 (1), 7 (2), 2 (3), 5 (4), 4 (5), 2 (6), 3 (7) } ans = 5

max = 5

max = 7

Q3. { 10 (0), 7 (1), 9 (2), 3 (3), 2 (4), 4 (5) } ans = 3.

max = 4

max = 9

Q4. { 8, -2, 4, 7, 6, 5, 1 } ans = 5

max = 7

Q.) { 8, 8, 8, 8 }. ans = 1

Brute-force Idea.

For every element, find max on right.

if that element > max on right

→ increment your count.

```
for( i = 0 ; i < N ; i++){
    //find max on r.h.s
    for( j = i+1 ; j<N ; j++){
    {
    }
}
```

T.C → O(N²)
S.C → O(1)

Optimisation: Carry forward max value from right to left.

max = 1 3 6 7 8

leaders = 1 2 3 4 5

arr → { 8 (0) -2 (1) 4 (2) 7 (3) 6 (4) 5 (5) 1 (6) }

pseudo.code:

```
max = arr[N-1] , leaders = 1
for( i = N-2 ; i >=0 ; i--) {
    if ( arr[i] > max ){
        leaders += 1
        max = arr[i]
    }
}
return leaders
```

TC → O(N)
S.C → O(1)

Sub-array Basics

definition → continous part of an array.

(a) A single element is a sub-array.

(b) Full array is also a sub-array.

(c) [-], it is not a sub-array.

empty array

Eg: arr[9] : { -3 4 6 2 8 7 14 9 21 }
0 1 2 3 4 5 6 7 8

Indices : [2, 3, 4, 5] : Yes.

Indices : [3, 4, 6, 7, 8] : NO

8, 0, 1, 2. : No.

indices : [1, 2, 3] : YES

indices : [5] : YES

indices : [2, 7] → 6 elements.

[a, b] → b - a + 1. [length of sub-array from a to b]

from now on, you can use some pre-defined functions-

Min (a, b)
Max (a, b)
] → {O(1) T.C.}

Sort (arr)] → O(NlogN)

Sorting → arranging data in asc/desc order

Q: Minimum element in the array.

```
ans = arr[0]
for( i = 1 ; i < N ; i++){
    ans = min( ans, arr[i])
}
return ans
```

T.C → O(N)

Closest Min-Max

Given an array. find the length of smallest subarray which contain both min & max of an array.

min = 1 , max = 6

Eg: { 1 2 3 1 3 4 6 4 6 3 }
0 1 2 3 4 5 6 7 8 9

subarray [3-8] : len = 8 - 3 + 1 = 6
[3-6] : len = 6 - 3 + 1 = 4
[0-6] : len = 6 - 0 + 1 = 7

ans = 4.

Q: { 2 2 6 4 5 1 5 2 6 4 1 }
0 1 2 3 4 5 6 7 8 9 10

min → 1
max → 6

ans = 3

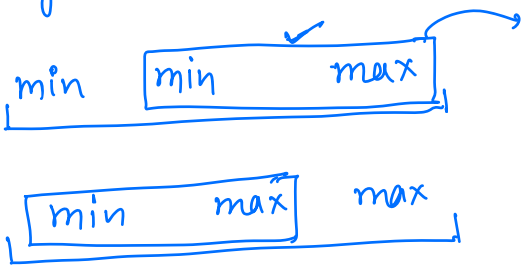
Q: [8 8 8 8 8]
0 1 2 3 4

min → 8
max → 8

ans = 1

Observations for final ans sub-array

① We only need to have 1 min & 1 max in our ans sub-array.



② Min & max should be the end-points of the sub-array.

[- - - [min - - - max] - -]

③

Case 1 : [min - - - max] → find closest max element position of on r.h.s.

Case 2 : [max - - - - - min] → find closest min element position of on r.h.s.

Eg: min = 1 max = 6

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

:[2,5] len = 4

:[5,8] len = 4

:[8,10] len = 3

no update

} min_len = 3.

Pseudo-code.

```
// find minVal and maxVal
if (minVal == maxVal) { return 1 }

ans = N
for( i = 0 ; i < N ; i++) {
    if (arr[i] == minval) {
        for( j = i+1 ; j < N ; j++){
            if (arr[j] == maxval){
                ans = min( ans , j-i+1)
                break
            }
        }
    }
    else if ( arr[i] == maxVal ) {
        for ( j = i+1 ; j < N ; j++){
            if (arr[j] == minval){
                ans = min( ans , j-i+1)
                break
            }
        }
    }
}
return ans
```

find closest maximum element on r.h.s

find closest minimum element on r.h.s

T.C → $O(N^2)$

S.C → $O(1)$

idea → Carry forward maxidx and minidx from r.h.s.

[min = 1 , max = 6.]

ans = N
minidx = -1
maxidx = -1

arr →	1 ✓	6 ✓	4 ✗	6	5 ✗	1	5 ✗	2 ✗	6	4 ✗	4 ✗	2 ✗	1 ✓	5 ✗
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

mini = 0
max = 1
[0,1] : 2
ans = 2.

maxi = 1
mini = 5
[1,5] : 5
ans = 1

maxi = 3
mini = 5
[3,5] : 3
ans = 3

mini = 5
maxi = 8
[5,8] : 4
ans = 4

maxi = 8
mini = 12
[8,12] : 5
ans = 5

mini = 12
maxi = -1
→ no update.

↑ ans.

⇓

len = maxi - mini + 1

minVal = 1
maxVal = 6

pseudo.code -

① find minVal & maxVal.

② if (minval == maxVal) return 1

③ ans = N , mini = -1 , maxi = -1

// -1 denotes, we dont have min/max value as of now.

④.
```
for ( i = N-1 ; i >= 0 ; i--) {
    if (arr[i] == minval) {
        mini = i
        if ( maxi != -1 ) {
            ans = min ( ans , maxi - mini + 1)
                                |maxi - mini| + 1
        }
    }
    else if (arr[i] == maxval) {
        maxi = i
        if ( mini != -1 ) {
            ans = min ( ans , mini - maxi + 1);
        }
    }
}
```

⑤ return ans;

T.C → O(N)

S.C → O(1)

todo. → traverse from left to right. {personal}.

4th-5th October → Dushera.