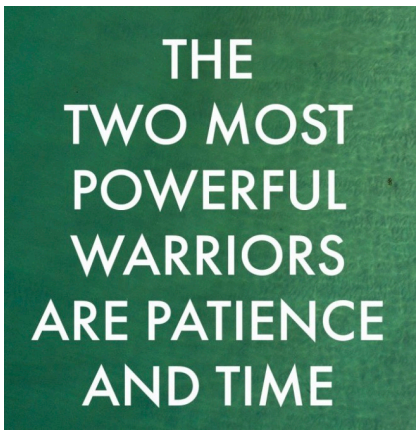
Today's Quote -



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.

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

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

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 the pairs.

```
count = 0 ;
for( i → 0 to n-1) {
    for( j → i+1 to n-1) {
        if (s[i] == 'a' && s[j] == 'g'){
            count += 1 ;
        }
    }
}
```

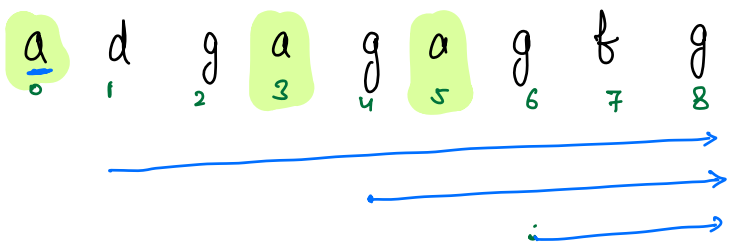
T.C → $O(N^2)$, S.C → $O(1)$

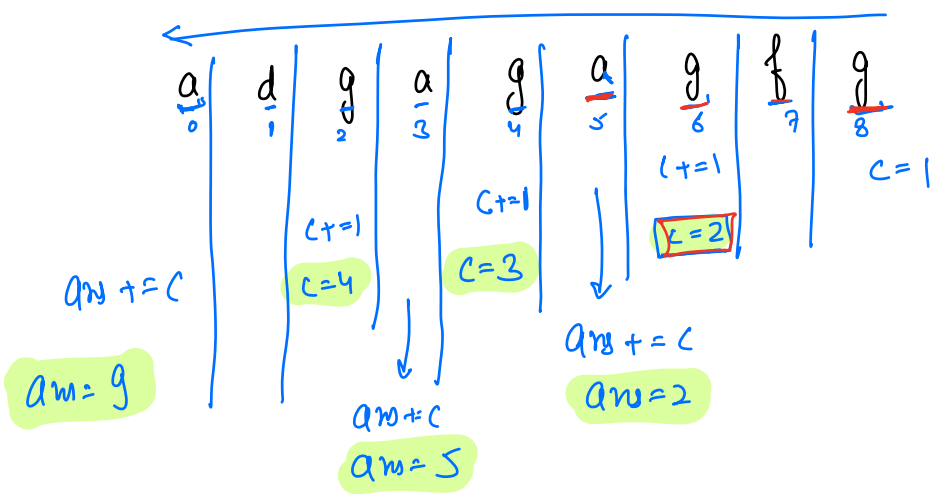
idea 2. : Consider only when s[i] = 'a'.

```
count = 0
for( i → 0 to n-1) {
    if (s[i] == 'a') {
        for( j → i+1 to n-1) {
            if (s[j] == 'g') {
                count += 1
            }
        }
    }
}
```

T.C → $O(N^2)$

Eg:





maintain count of g's from r.h.s.

pseudo code:

```
count = 0 , ans = 0 ;

for( i → N-1 to 0 ){
        if (s[i] == 'g'){
            count += 1
        }
        else if (s[i] == 'a'){
            ans += count
        }
}

return ans ;
```

T.C → O(N)
S.C → O(1)

Alternative approach.

For every 'g', find the no. of a's present on left.s .

T.C → O(N)
S.C → O(1)

todo: Traverse the array from left to right Carry forward the count of a's.

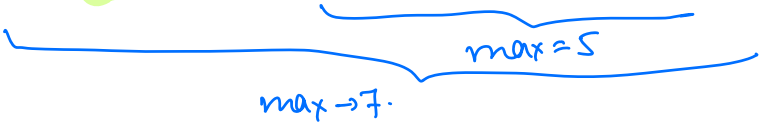
Q2.) **Leaders in An Array**

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

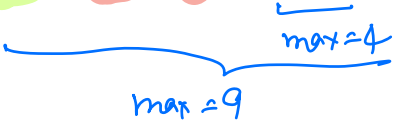
An element is leader, if it is strictly greater than max on right side.

Note. → arr[N-1] is always considered as leader.

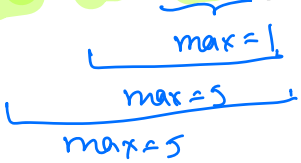
Eg: { 15 (0), -1 (1), 7 (2), 2 (3), 5 (4), 4 (5), 2 (6), 3 (7) } ans = 5



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



Q4. { 8, -2, 4, 7, 6, 5, 1 } ans → 5



Q.) { 8, 8, 8, 8 }.

Brute-force Idea.

For every element, find max on right.
if that element > max on right
increment your count.

```
for( i → 0 to N-1) {
    // find max on right
    for( j → i+1 to N-1) {
        ==========
    }
}
```

T.C → O(N²)
S.C → O(1)

Optimisation: Carry forward max value from right to left.

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

max → 1 ~~5~~ ~~6~~ 7 8
ans → 1 ~~2~~ ~~3~~ 4
5

pseudo code.

```
max = arr[n-1] , ans = 1
for( i → n-2 to 0) {
    if(arr[i] > max) {
        ans += 1
        max = arr[i]
    }
}
return ans
```

T.C → O(n)
S.C → O(1)

Sub-array Basics

definition → continuous 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

indices : [1, 2, 3] : YES

indices : [5] : YES

Indices : [2 - 8] → 7 elements [8 - 2 + 1]

length of sub-array :

[a, b] → [b - a + 1]

Break till 10 : 40

Eg { 4, 5, 1, 9, 0, 2, 3, 5 }

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

- min(a, b) → Math.min(a, b), minimum(a, b)
- max(a, b) → O(1) T.C

- sort(arr) ⇒ sort, arrange data in ascending order.

O(N log N) T.C

// Given arr[]. find min in the array.

```
ans → arr[0];
for( i → 1 to n-1) {
    ans = min( ans, arr[i])
}
return ans;
```

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

Subarray: [3-8] : len = 8-3+1 = 6 } ans = 4
[3-6] : len : 6-3+1 = 4
[0-6] : len : 7

Min → 1
Max → 6

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

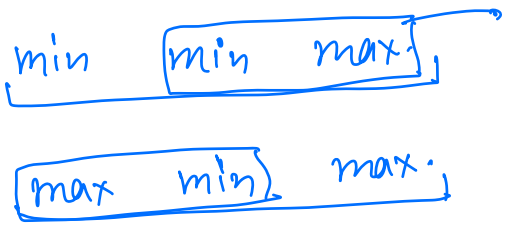
[ans → 3.]

Quiz. { 8 8 8 8 8 8 } [ans = 1]
idx: 0 1 2 3 4 5

Min = 8
Max = 8

Observations for final ans sub-array

① We only need to have 1 min & 1 max in our subarray.

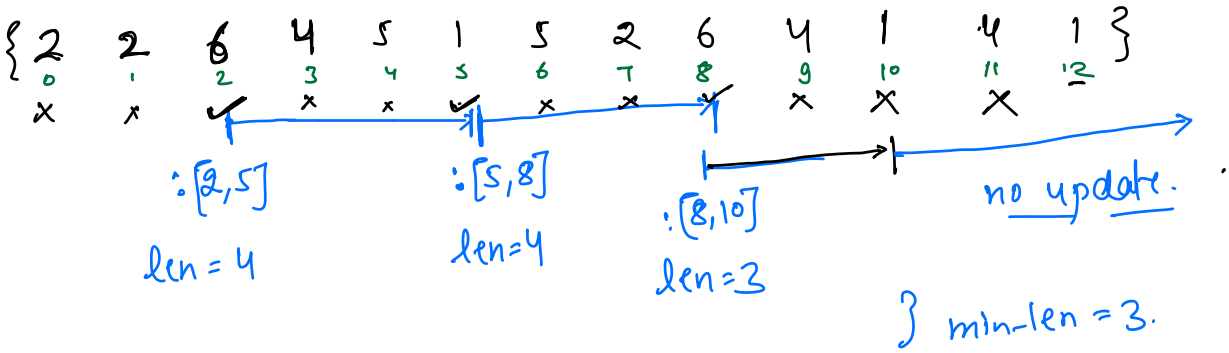


② Min & max should be end points of the subarray.

③ Case 1 : [Min .... Max] → find closest max element on r.h.s.

Case 2 : [Max .... Min] → find closest minimum element 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 to n-1) {
    if (arr[i] == min-Val) {
        for (j = i+1 to n-1) {
            if (arr[j] = maxVal) {
                ans = min (ans, j-i+1)
                break
            }
        }
    }
    if (arr[i] == maxVal) {
        for (j = i+1 to n-1) {
            if (arr[j] == minVal) {
                ans = min (ans, j-i+1)
                break
            }
        }
    }
}
return ans;
```

find closest max value on r.h.s

find closest min value on r.h.s.

T.C → $O(N^2)$

S.C → $O(1)$

idea → carry forward maxidx and minidx.

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
x x x x x x x

mini = 0
maxi = 1
(0,1) : 2
ans = 2

maxi = 1
mini = 5
(1,5) : 5
ans = 3

maxi = 3
min = 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

2

✓ maxi = -1
✓ mini = -1

pseudo-code -

① find minval and maxVal by traversing in array.

② if (minval == maxVal) return 1

③ ans = N , maxi = -1 , mini = -1 // -1 denotes min, max not found.

④
```
for ( i → N-1 to 0) {
    if arr[i] == minval) {
        mini = i ;
        if (maxi != -1) {
            ans = min (ans, maxi-mini+1)
        }
    }
    else if ( arr[i] == maxVal) {
        maxi = i
        if (mini != -1){
            ans = min( ans, mini-maxi+1)
        }
    }
}
return ans
```

T.C → O(N)
S.C → O(1)

to-do (traverse from left to right)

```
mini != -1 &&
if (mini - maxi + 1 < ans) {
    ans = mini - maxi + 1
}
```

Doubts :

A → [1, 2, 3, 4, 5]
0 1 2 3 4

prefix product = [ ]

suffix product [120, 120, 60, 20, 5]

for(i = n-2 to 0) {

{ spro[i] = arr[i] * spro[i+1] }

}

if (arr[0] % 2 == 0) {

pSum[0] = 1

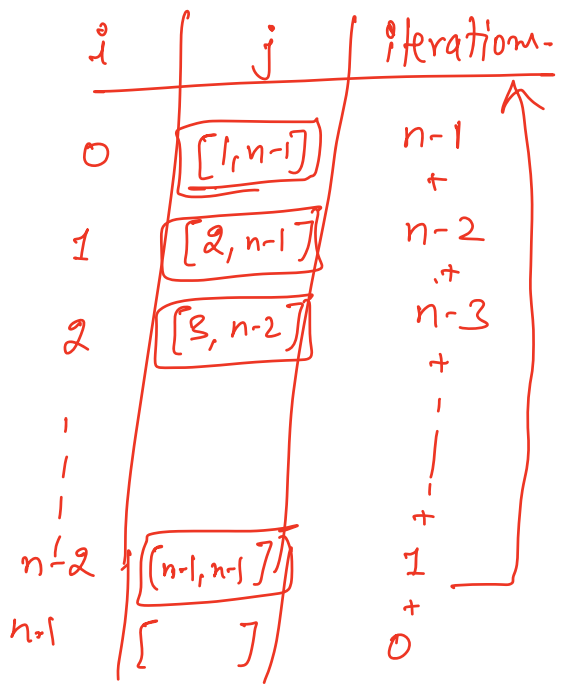
}

else {

pSum[0] = 0

}

```
for( i = 0 ; i < n ; i++){
    for( j = i+1 ; j < n ; j++){
        //
    }
}
```



total no. of iterations = $1+2+3+\cdots+(n-2)+(n-1)$

$= \frac{n(n-1)}{2}$.

```
for( i = 0 ; i < arr.length ; i++){
    if (arr[i] % 2 == 0){
        arr[i] = 1
    }
    else {
        arr[i] = 0
    }
}
```