Welcome ☺

Agenda : Carry Forward Technique.

3 problems

Q1 Given a string of lowercase letters, return count of (i, j) such that i<j and s[i] is 'a' and s[j] is 'g'.

eg : s : "abegag"

count = 3 → (0, 3) (0, 5) (4, 5)

Quiz 1 acgdgag

count = 4 → (0, 2) (0, 4) (0, 6) (5, 6)

Quiz 2 bcaggaag

count = 5 → (2, 3) (2, 4) (2, 7) (5, 7) (6, 7)

Brute force:

For every 'a', count all 'g' on right hand side.

Pseudo code

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

T.C → $O(N^2)$

S.C → $O(1)$

Obs 1

We will store count of 'a' and whenever 'g' is encountered, we will add the count of 'a' to the result.

eg :	a	c	b	a	g	k	a	g	g	a
count_a	1	1	1	2	2	2	3	3	3	4
total	0	0	0	0	2	2	2	5	8	8

Optimised

```
total = 0
count_a = 0;
for ( int i= 0 ;  i< N ;  i++)
{
    if( S[i] = = 'a')
    {
        count_a += 1 ;
    }
    if ( s[i] = = 'g')
    {
        total = total + count_a ;
    }
}
return total ;
```

T.C → O(N)

S.C → O(1)

→ This approach is called as carry forward.
We are maintaining running count.

Amazon Zeta

Given an array, return the length of the smallest subarray which contains both the max, min of the array.

Sub question

How many subarrays are there in an array of size N

Count of subarray from index 0 → N

Count of subarray from index 1 → N-1

⋮

Count of subarray from index N-1 → 1

Total count of subarrays ⇒ N + N-1 + N-2 . . . . . . + 1

$$\Rightarrow \frac{N(N+1)}{2}$$

eg:

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

max = 6

min = 1

ans = 4 [3, 6] index

eg:

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

max = 6

min = 1

[8,10]

ans = 10-8+1 = 3

Brute force

→ Check all subarrays. Check if they max & min and then compare the length.

T.C → $O(N^3)$

↓

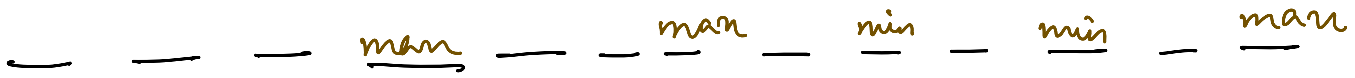
$O(N^2)$

obs 1 The ans. subarray must have only one max & min.

obs 2 The min and max in the ans subarray will be present at edges.

[max . . . min]

[min _ _ _ _ _ max]



eg: { 2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1 }

last min Index = -1

last max Index = -1

ans = INT_MAX

min Value = 1

max Value. = 6

i	A[i]	lastminI	lastMaxI	ans
0	2	-1	-1	INT_MAX
1	2	-1	-1	"
2	6	-1	2	"
3	4	-1	2	"
4	5	-1	2	"
5	1	5	2	5-2+1 = 4
6	5	5	2	"
7	2	5	2	"
8	6	5	8	4
9	4	5	8	4
10	1	10	8	10-8+1 = 3

Optimised

```
// Find max & min of entire array.

lastMaxI = -1
lastMinI = -1
     ans = N

for( int i=0 ; i<N ; i++)
{
    if( A[i] == A_min)
    {
        lastMinI = i
        if( lastMaxI >= 0)
        {
            ans = min( ans, i- lastMaxI +1)
        }
    }
    if( A[i] == A_max)
    {
        lastMaxI = i ;
        if ( lastMinI >= 0 )
        {
            ans = min ( ans , i- lastMinI + 1)
        }
    }
}
return ans;
```

T.C → O(N)
S.C → O(1)

Q3 Given an array of size N, count the number of leaders.

leader : Any element that is greater than all the elements on the left side.

$A[i] > [0 \dots i-1]$

eg : 2 5 3 4 17 16

Ans- 3 (2, 5, 17)

index '0' is always a leader.

Brute force

2 loops.

T.C → $O(N^2)$

Outer loop → 0 → N-1

Inner loop → 0 → i-1

Compare all elements on left side

Optimised

⇒ Keep track of maximum values while traversing the array.

Pseudo code

```
int count = 1
int lastMax = A[0];
for( i = 1 ; i< N ; i++)
{
    if( A[i] > lastMax)
    {
        count ++ ;
        lastMax = A[i] ;
    }
}
return count ;
```

T.C → $O(N)$

S.C → $O(1)$