Today's Agenda

→ Carry forward & Subarray.

Contest → End of Module.

1.5 hrs → 3 Questions
↓
2 Questions.

Ques

Given a string **s** of lowercase characters, return the **count of pairs (i,j)** such that **i < j** and **s[i] is 'a'** and **s[j] is 'g'**.

s = 'abegag' Ans = 3.

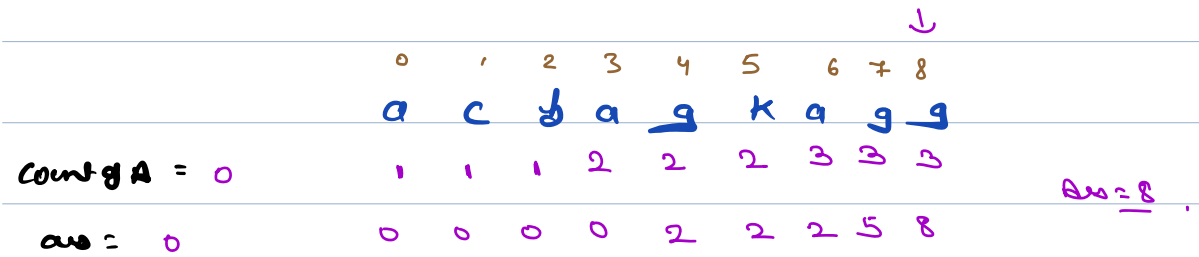
s = 'acgdgag' Ans = 4.

s = 'bcaggaag' Ans = 5

Brute force :-

```
result = 0;                                   T.C -> O(N^2);
for (i=0; i<n; i++) {                         S.C -> O(1);
    if (s[i] == 'a') {
        for (j=i+1; j<n; j++) {
            if (s[j] == 'g')
                result++
        }
    }
}
return result;
```

* Optimized Idea :-



Ans = 8.

```
result = 0;
countA = 0;
for(i=0; i<n; i++) {
    if(str[i] == 'a') {
        countA++;
    }
    else if(str[i] == 'g') {
        result += countA;
    }
}
return result;
```

T.C → O(n)

S.C → O(1)

Introduction to Subarrays

A subarray is a contiguous part of an array. It is formed by selecting a range of elements from the array. A subarray can have one or more elements and must be a contiguous part of the original array.

0	1	2	3	4	5	6	7	8
4	1	2	3	-1	6	9	8	12

[4, 1, 2] → ✓

[2, 3, -1, 6] → ✓

[9] → ✓

[4, 1, 2, 3, -1, 6, 9, 8, 12] → ✓

[4, 12] → ✗

[1, 2, 6] → ✗

[3, 2, 1, 4] → ✗

Which of the followings is a subarray of [4,5,1,9,0,2,3,5]?

a) [5] ✓

b) [4, 5, 1, 0] ✗

c) [9, 0, 2, 3] ✓

d) [4, 5, 1] ✓

← Represent a subarray →

0	1	2	3	4	5	6	7	8
4	1	2	3	-1	6	9	8	12

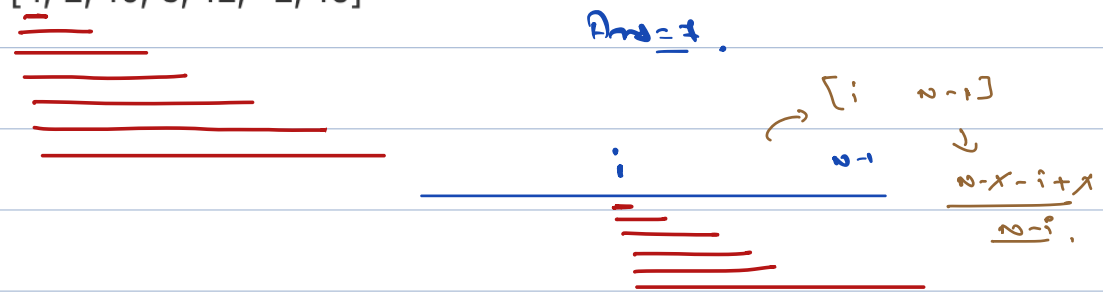
Two ways

1) Start & end idx. -> [2 5]

2) start idx & len →
   ↓2 ↓4

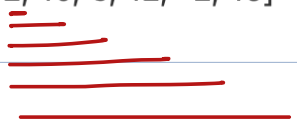
How many subarrays of the following array start from index 0

[4, 2, 10, 3, 12, -2, 15]



How many subarrays of the following array start from index 1

[4, 2, 10, 3, 12, -2, 15]



$[\ a_0\ \ a_1\ \ a_2\ \ \ldots\ \ a_{n-2}\ \ a_{n-1}\ ]$ (len n).

Subarrays starting from 0^{th} idx $\rightarrow n$

Subarrays starting from 1^{th} idx $\rightarrow n-1$

Subarrays starting from 2^{th} idx $\rightarrow n-2$

$\vdots$

Subarrays starting from $n-2^{th}$ idx $= 2$

Subarrays starting from $n-1^{th}$ idx $= 1$

Total no of subarrays of an array $\rightarrow \frac{n(n+1)}{2}$

[10, 20, 30] → $3\left(\frac{3+1}{2}\right)$ => 6.

[0 0] => 10

[0 1] = 10, 20

[0 2] = 10, 20, 30

[1 1] = 20

[1 2] = 20, 30

[2 2] = 30

Break 8:02 AM - 8:12 AM

Ques startIdx → endIdx

0	1	2	3	4	5	6	7	8
4	1	2	3	-1	6	9	8	12

```
for (i = startIdx ; i <= endIdx; i++) {
    print(arr[i])
}
```

T.C → O(N)
S.C → O(1).

Q) Print all subarrays:-

[10, 20, 30] (0, 1, 2) → $\frac{3(3+1)}{2}$ => 6.

[0 0] => 10

[0 1] = 10, 20

[0 2] = 10, 20, 30

[1 1] = 20

[1 2] = 20, 30

[2 2] = 30

10, 20, 30

[10, 20, 30] (0, 1, 2)

→ T.C → $O(N^3)$.

S.C → O(1).

start	end
0	0
0	1
0	2

```
for (start=0; start<n; start++){
    for (end=start; end<n; end++) {
        for (i=start ; i<=end ; i++){
            print(arr[i])
        }
        print("\n");
    }
}
```

Max & Min

Given an array of N integers, return the length of smallest subarray which contains both maximum and minimum element of the array.

Max = 6
Min = 1

2	2	6	4	5	1	5	2	6	4	1

Ans = 2.

Max = 6
Min = 1

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

A[ ] = { 1, 2, 3, 1, 3, 4, 6, 4, 6, 3}

Ans = 4.

Brute force (check all subarrays)

→ first find Max and Min → O(N).

→ checking all subarray → O(N^3)

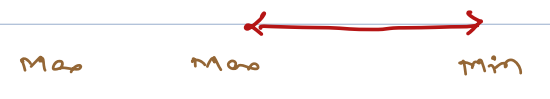
T.C → O(N^3)

S.C → O(1).

Max = 6
Min = 1

```
          0  1  2  3  4  5  6  7  8  9
A[ ] = { 1, 2, 3, 1, 3, 4, 6, 4, 6, 3}
```

obs 1 : in the any window there will be one maxm & one minm element present.



obs 2 :- Max & Min elem will be corner elements of your window.

- So, basically we are looking for subarray **which starts with maximum value and ends with closest minimum value or which starts with minimum value and ends with closest maximum value.**

```
for (i=0; i<n; i++) {
    if (arr[i] > max) {
        max = arr[i]
    }
}
```

max & min.

last Max i

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

A[ ] = { 2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1 }

Max = 6
min = 1
last_Min_Idx = ~~-1~~ ~~5~~ 10
last_Max_Idx = ~~-1~~ ~~2~~ 8
ans = ~~∞~~ ~~4~~ 3

2 2 1 4 5 6

Max

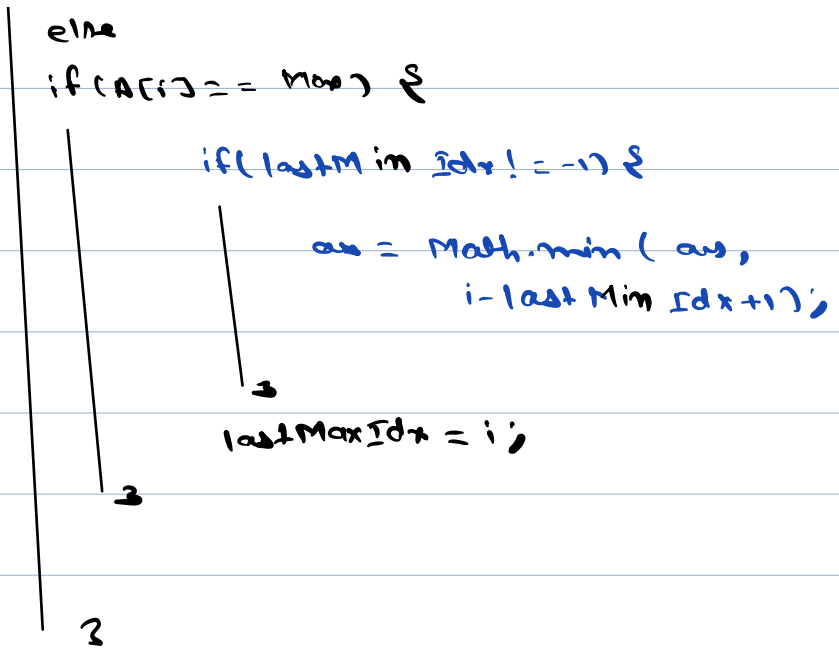
new Max Idx = -1.

```
int min = Min(arr);
int max = Max(arr);
int lastMinIdx = -1;
int lastMaxIdx = -1;
int ans = ∞;
```

T.C → O(N)

S.C → O(1).

```
for(i=0; i<n; i++) {
    if(A[i] == min) {
        if(lastMax Idx != -1) {
            ans = Math.min( ans,
                i - lastMaxIdx + 1);
        }
        lastMinIdx = i;
    }
```



return ans;

Note:- Prefix sum is used when we are performing range queries.
In other words, we have to calculate subarray sum repeatedly.

Carry forward:- when we can use the result previously calculated for the next idx, instead of recalculating it again from the start,

i

[x, y, z, a, b, c]

(0 - (i-1)) (i+1 - N-1)

pf[i-1] = pf[N-1] - pf[i]

[1, 2, 5, 7, 9]

pfn → [1, 3, 8, 15, 24]

sum[i] = pf[i] - pf[i-1] (i!=0)

sum = x

sum1 = x + y

sum2 = x - x - y - z

↓

[x, y, z, a, b, c]

↓

0 1 2 3 4 5 6

[-7, 1, 5, 9, -4, 3, 0]

pf [-7, -6, -1, 8, -3, 0, 0]

(0-2) (4-6)

pf[2] pf[6] - pf[3]

-1 1 - 0