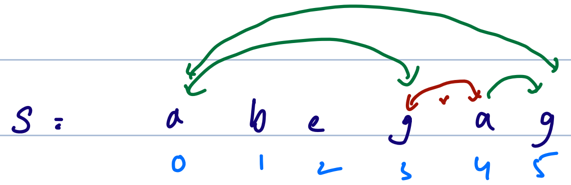
Question: Given a string of lowercase alphabets, return the count of pairs[indices] (i, j) such that i < j and s[i] = 'a' and s[j] = 'g'

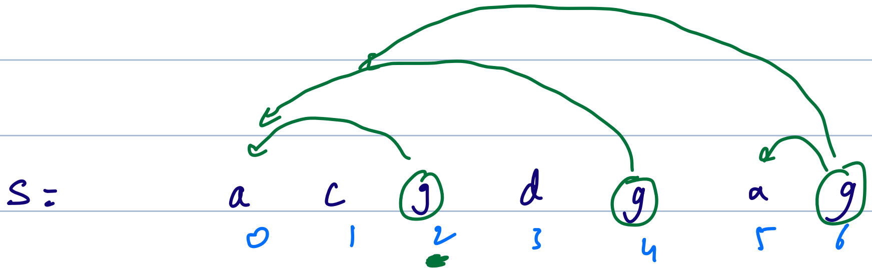
$len(s) \le 10^6$



(0, 3)

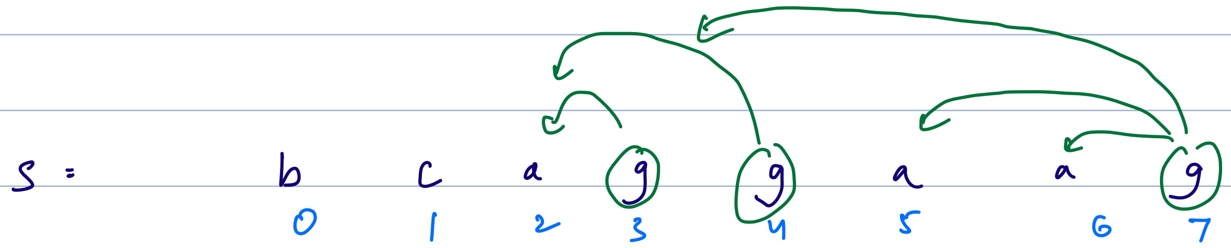
(0, 5)

(4, 5)



(0, 2) (0, 4) (0, 6) ⇒ 4

(5, 6)



(2, 3) (2, 4) (2, 7)

(5, 7) ⇒ 5 pairs

(6, 7)

Brute Force

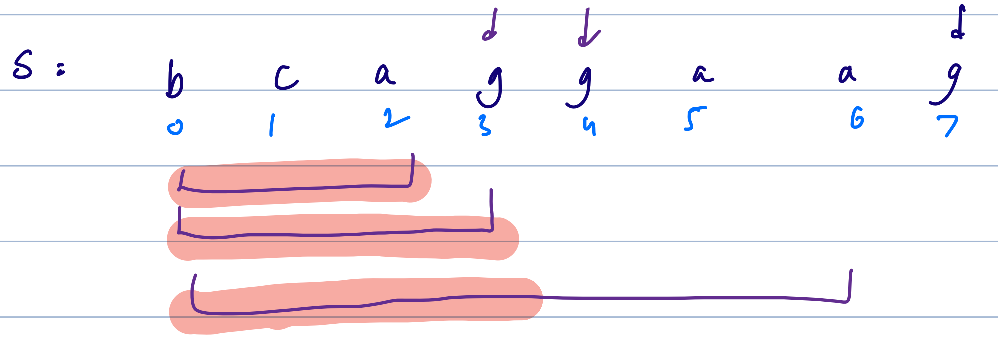
Iterate the string, for every 'g' count the no. of pairs it forms by counting # a's to its left

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

T.C: $O(N^2)$

S.C: $O(1)$

Efficient:

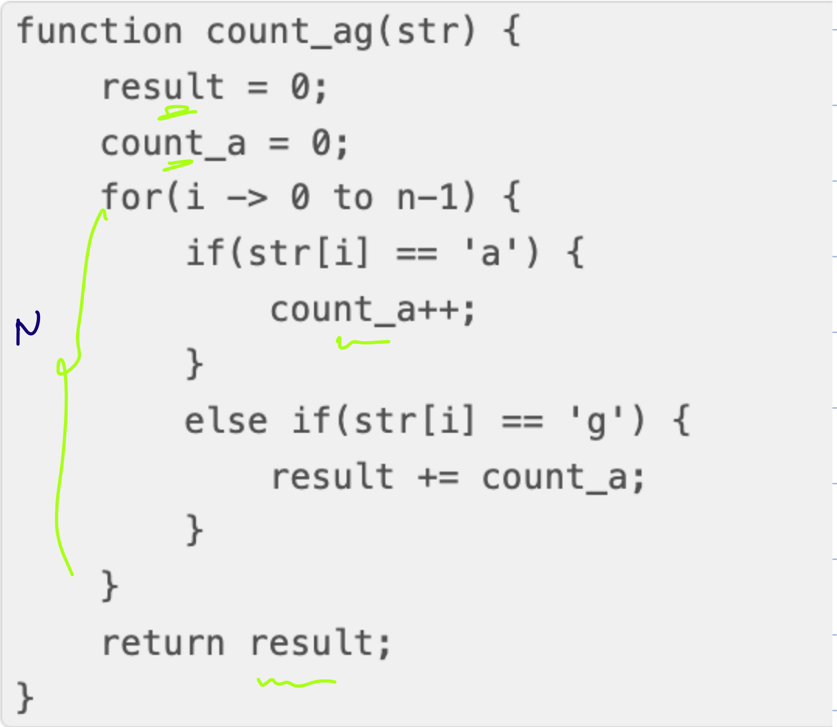


S :

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

result = $\cancel{0} + 1 + 1 + 3 = \boxed{5}$

count_a = ~~0~~ ~~1~~ ~~2~~ 3



N

```
function count_ag(str) {
    result = 0;
    count_a = 0;
    for(i -> 0 to n-1) {
        if(str[i] == 'a') {
            count_a++;
        }
        else if(str[i] == 'g') {
            result += count_a;
        }
    }
    return result;
}
```

T.C: O(N)

S.C: O(1)

CARRY FORWARD

SUBARRAY

-) A contiguous part of an array

-) Single element can also be a subarray

-) Entire array is a subarray

A :	3	4	5	6	-2	8	10
	0	1	2	3	4	5	6

[5, 6, -2] ✓ => (2, 4)

[3, 4, 6, -2] ✗

[3, 4, 5, 6, -2] ✓ => (0, 4)

[5, 4, 3] ✗

[5] ✓ (2, 2)

[3 4 5 6 -2 8 10] ✓ ( )

Quiz : 2 4 1 6 -3 7 8 4

How to represent a subarray ?

(Start Index, end Index)

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

0 1 2 3 4 5 6

start = 0

end : [0, 6] => 6-0+1 = 7

(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6)

7 subarrays

Quiz :

4	2	10	3	12	-2	15
0	1	2	3	4	5	6

N = 7

start = 1

end = [1, N-1] => [1, 6] => $6-1+1 = \boxed{6}$

Question: Total No. of Subarrays

end

#subarrays starting at index 0 => [0, N-1] : N

+

#subarrays starting at index 1 => [1, N-1] : N -1

+

#subarrays starting at index 2 => [2, N-1] : N - 2

+

#subarrays starting at index 3 => [3, N-1] = N - 3

+

⋮

#subarrays starting at index N-1 [N-1, N-1] : 1

$1+2+3+ \ldots \ldots N-2 + N-1 + N = \boxed{\frac{N(N+1)}{2}} \Rightarrow$

Qushon: Print a subarray

```
                                       Array
void  printSubarray( int  A[], int s, int e){
        for (i=s; i<=e; i++){
              print(A[i]);
        }
        print("/n");                        // Go to new line
}
```

T.C: $\boxed{O(N)}$

can we print an array is less than O(N)?

Queshon: Prnt all subarrays of an array

A = [1, 2, 3]   N = 3

$\quad\;\; 0 \quad 1 \quad 2$

$\frac{N(N+1)}{2} = \frac{3 \cdot 4}{2} = 6$

s	e		Subarray
0	0		[1]
0	1	=>	[1, 2]
0	2		[1, 2, 3]
1	1		[2]
1	2		[2, 3]
2	2		[3]

```
for(start = 0;    Start < N;   start++) {
        for(end = start;    end < N;   end++) {
              =>  // (start, end) represent a subarray
O(N) <-------  printSubarray ( A, start, end);
                                        Array
        }
}
```

T.C: $N^2 \times O(N) \Rightarrow O(N^3)$

S.C: $O(1)$

Question: Given an array, return the length of the smallest subarray (contiguous part of array) which contains both the min and max of the array

len = 8

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

len = 7

Max(A) = 6

Min(A) = 1

$\boxed{\text{len} = 4}$

$\boxed{8:32}$

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

Max(A) = 6

Min(A) = 1

len=3

Brute Force

=> Find Max and Min => O(N)

=> Iterate over all the $\frac{N(N+1)}{2}$ arrays, for each subarray, check it if has max and min, if yes update your answer

T.C: $O(N) + \frac{N(N+1)}{2} \times O(N) = O(N^3)$

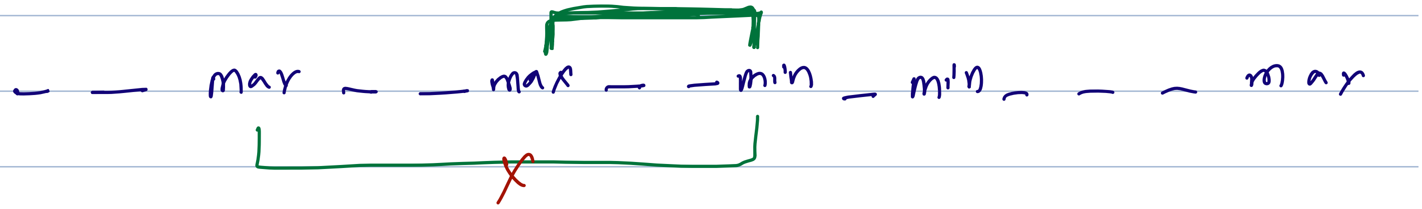
S.C: $O(1)$

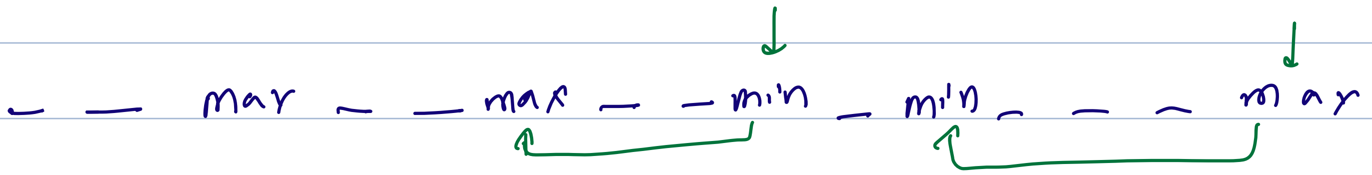
Approach2:

Observations:

1) the corner elements of the answer subarray are max and min

2) Answer subarray will have exactly 1 max and 1 min





-) Carry foward index of latest max and latest min.

-) when you get a min element, use the index of latest max

-) when you get a max element, use the index of latest min

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

Max(A) = 6

Min(A) = 1

latest_max = ~~-1~~ ~~2~~ 8

latest_min = ~~-1~~ ~~5~~ 10

Ans = ~~INT_MAX~~ ~~4~~ 3

(i - latest_max + 1)

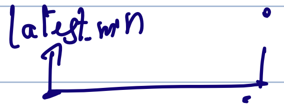
A : 4 4 4
0 1 2

latest_max = ~~-1~~ ~~0~~ ~~1~~ 2
latest_min = ~~-1~~ ~~0~~ ~~1~~ 2

ans = ~~INF~~ 1

```
// Find max (maxm.)
// Find min (minm)
latest_max = -1;
latest_min = -1;
ans = INFINITY;
for(int i = 0; i < n; i++){
          if(A[i] == maxm){
                    if(latest_min != -1){
     O(1)                     ans = min(ans, i - latest_min + 1);     ans = min(5, 3) = 3
                              ~~latest_max = i;~~
                    }
                    latest_max = i;
          }
          if(A[i] == minm){
                    if(latest_max != -1){
  O(1)                        ans = min(ans, i - latest_max + 1);
                              ~~latest_min = i;~~
                    }
                    latest_min = i;
          }
}
return ans;
```

T.C : O(N)
S.C : O(1)



len = i - latest_min + 1 =>

[a, b] = b - a + 1

	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
A:	2	2	6	4	5	1	5	2	6	4	1
	0	1	2	3	4	5	6	7	8	9	10

$$\left.\begin{array}{l}\text{Min}(A) = 1\\ \text{Max}(6) : 6\end{array}\right\}$$