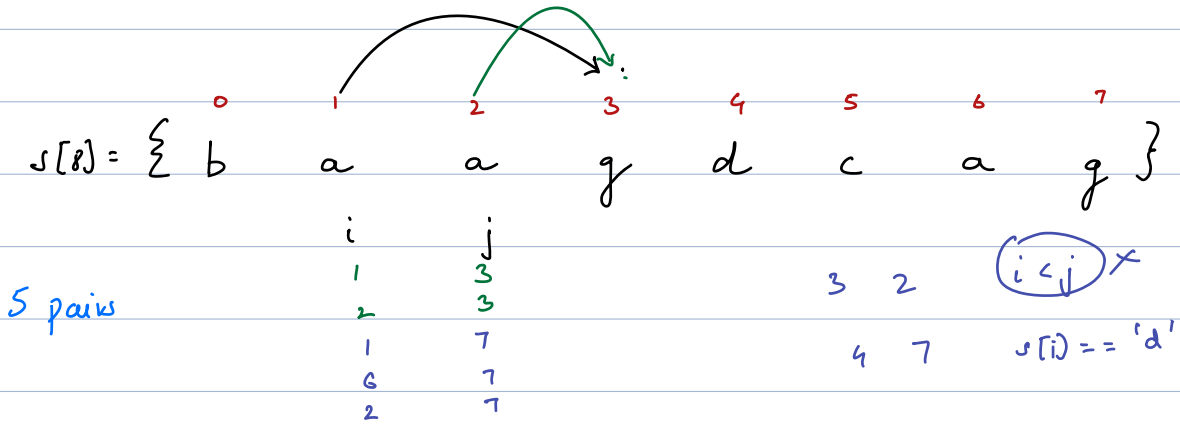
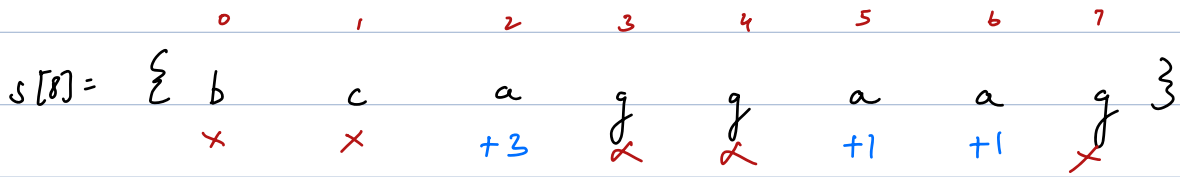
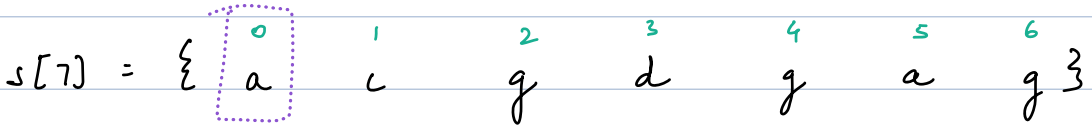
Q) Given a char [], calculate no. of pairs (i,j) such that $i < j$ && $s[i] = 'a'$ && $s[j] = 'g'$.

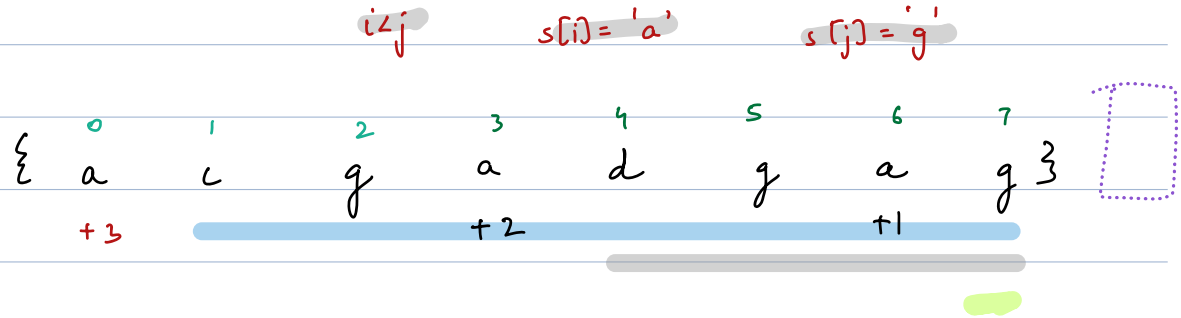




Ans : 5



Ans : 4



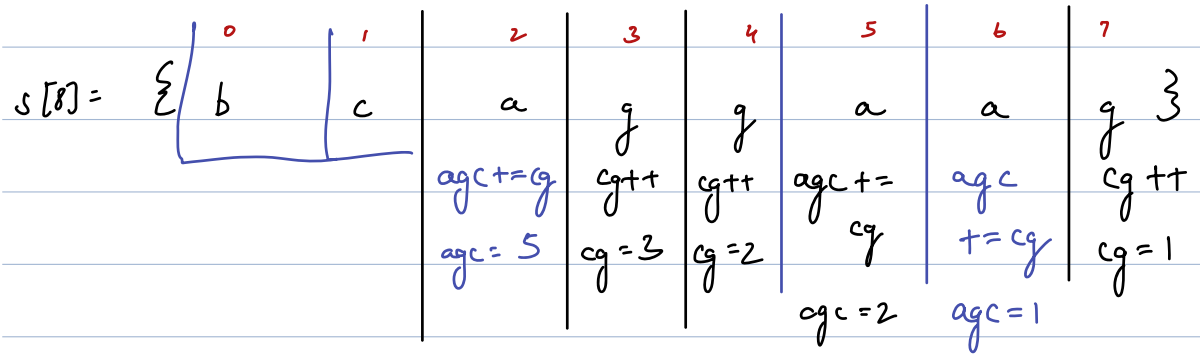
Ans: 6

TC: $O(N^2)$

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

Problem: For every a, we are repeatedly iterating the same array from L-R to count total no. of gs

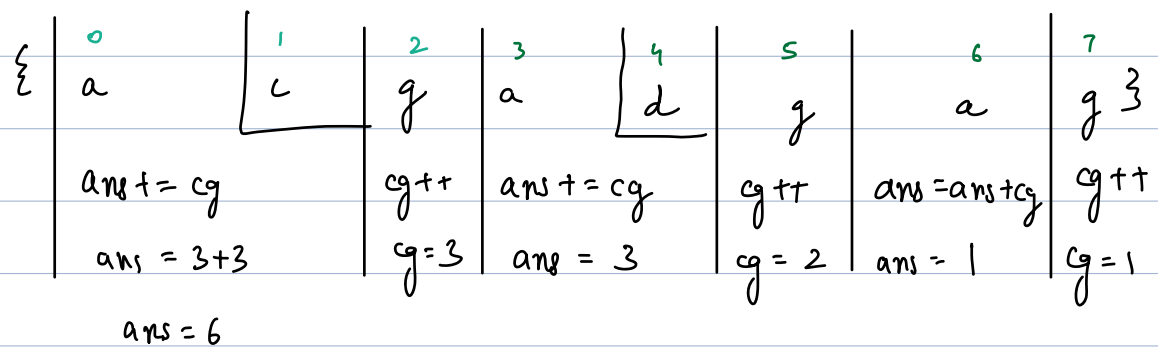
No. of gs on the right of every a



cg = 0 ag count = 0

ans = 0

agc → 5 cg = 0



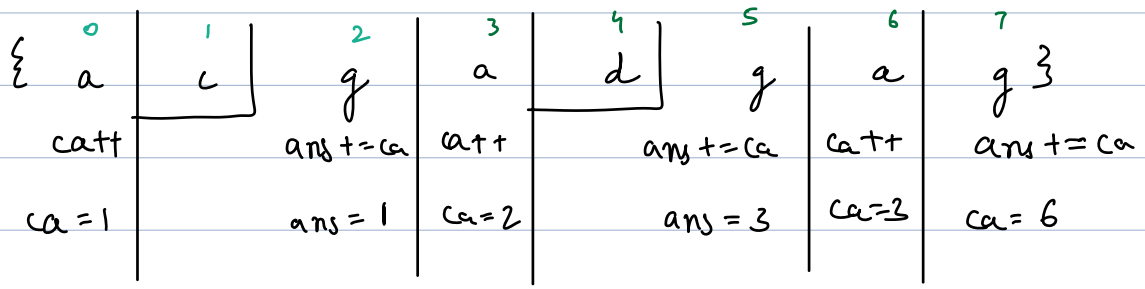
cg = 0 ans = 0

TC: O(N)

SC: O(1)

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

count a = 0 ans = 0



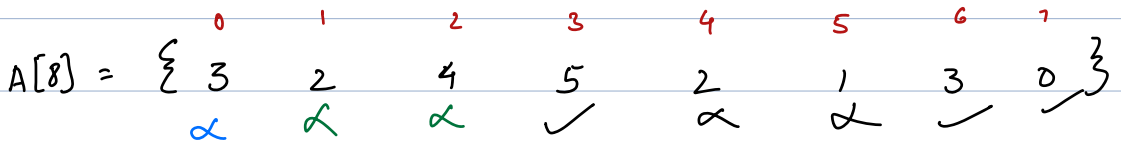
Miorosoft

Q2) Leaders in an Array

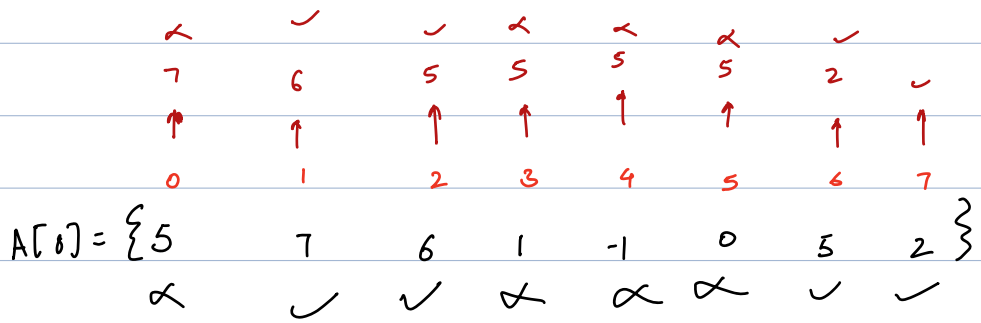
Given an array, you need to find leaders in A[]

An element is a leader if it is stricly greater than all elements on its right.

A[N-1] is always a leader



Ans: 3

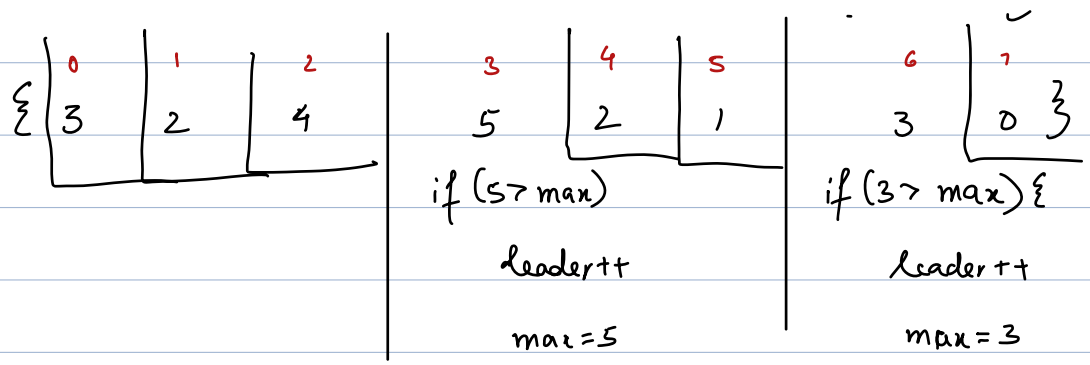


Ans: 4

If an element is greater than max of all elements on its right then it is a leader

```
for (i=0; i<N; i++) {
    # find the max on right        TC: O(N^2)
    # [i+1, N-1] → maxr            SC: O(1)
    if (A[i] > maxr) {
        leader++
    }
}
return leader
```

leaders = 3 max = 0



leaders = 1 max = A[N-1] TC: O(N)

```
for (i=N-2; i>=0; i--) {        SC: O(1)
    if (A[i] > max) {
        leader++
        max = A[i]
    }
}
return leader
```

	0	1	2	3	4	5	6	7
A[i] = {	7	7	6	1	-1	0	5	9
m = -∞	7	7	7	7	7	7	7	9

Max in array

min = ∞

Intge. min valu

inf

INT_MIN ✓

curmax = -∞

```
for (i=0; i<N; i++){
    if (A[i] > curmax){
        curmax = A[i]
    }
}
```

curmax = -∞

```
for (i=0; i<N; i++){
    curmax = max(curmax, A[i])
}
```

Break (10:32 - 10:42)

Subarrays continous part of array

→ Can single element be a subarray? Yes

→ Can entire array be a subarray? Yes

→ Can 0 elements be a subarray? No

{ 2 3 5 7 1 0 -3 }

{3} ✓

{ 2 3 5 7 1 0 -3 } ✓

{ 3 5 7 1 } ✓

{ 2 5 7 } ✗

Closest Min Max

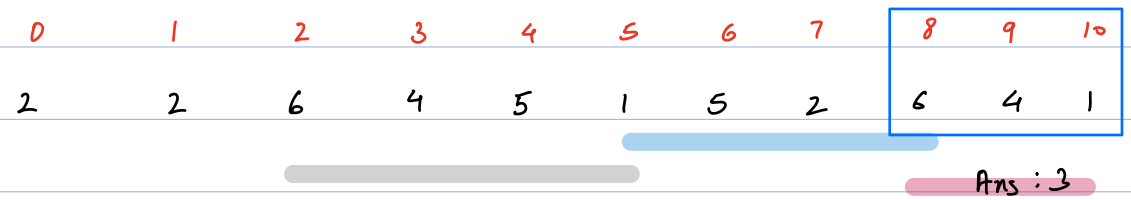
Amazon

Given an array find the length of smallest subarray which contains both min and max of array

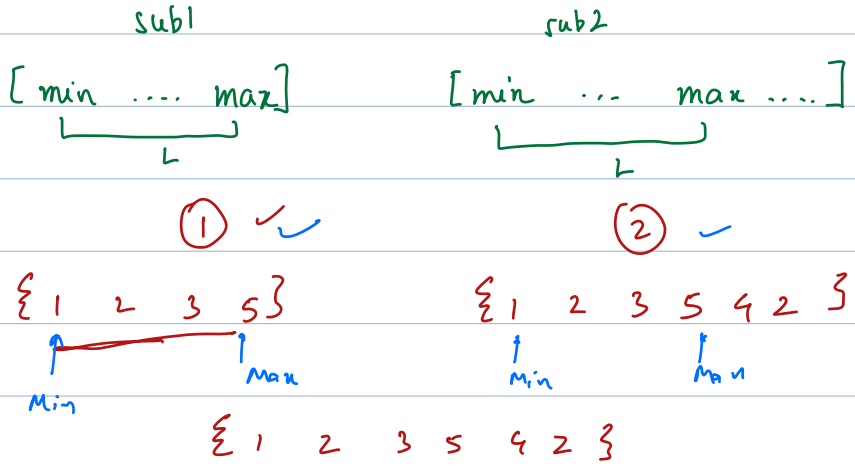
Min : 1 Max : 6

Ans : 4

Min : 1 Max = 6



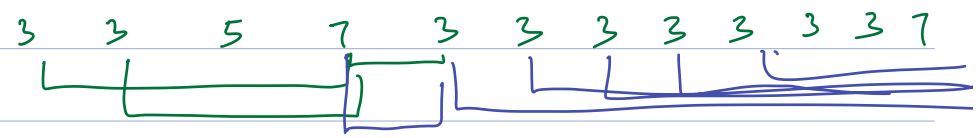
Obs 1) In the min len subarray, min ele & max ele will be at corners



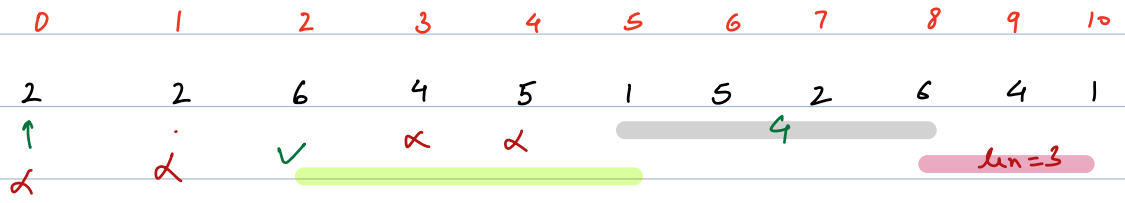
2)

[Min Max] [Max Min]

1) Find Min, Max



min = 1 , max = 6



[2, 5] = 5 - 2 + 1

= 4

-5

Max ⟶ Min

minlen = 4

Min ⟶ Max TODO:

1) Find and max TC: $O(N^2)$

min len = $+\infty$ SC: $O(1)$

iterate over all the elements

→ if it is not min or max

skip

→ if it is max

l = i

look for min index on right [i+1, N-1] → r

len = r - l + 1

minlen = min (minlen, len)

→ if it is min

l = i

look for max index on right [i+1, N-1] → r

len = r - l + 1

minLen = min (minLen, len)

1) Min, Max values → O(N)

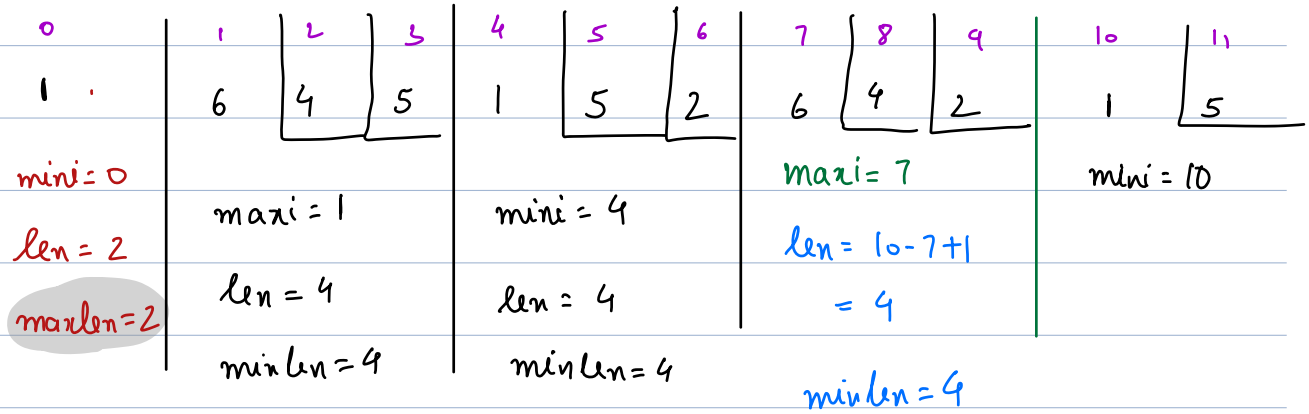
2) mini = -1 maxi = -1

3) minLen = ∞ / N

Min → max on right

Max → min on right

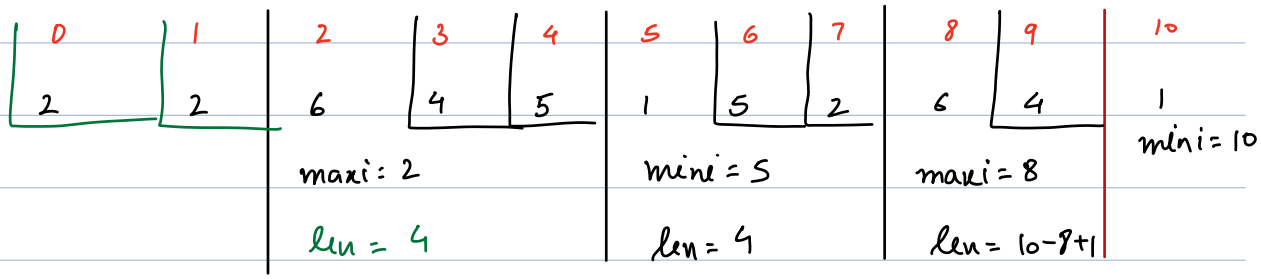
max = 6 min = 1

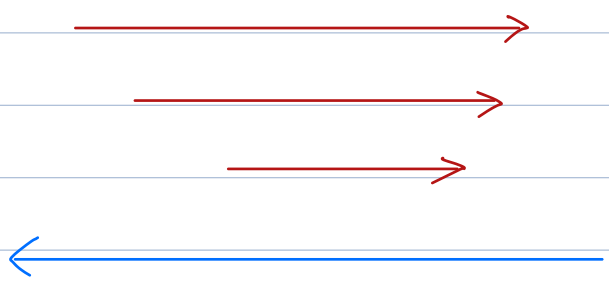


[2, 2, 2, 2]

min = 1 max = 6

maxi = -1 mini = -1 minlen = ∞/N

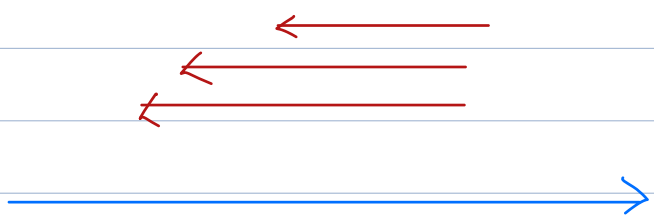




r - l + 1

↑

r > l



maxi = -1 , mini = -1 , minlen = ∞

1) min-v max-v

```
if (maxv == minv){
}
return 1
```

```
for (i = n-1 ; i >= 0 ; i--) {
    if (A[i] == max) {
        maxi = i
        if (mini != -1) {
            len = max(mini, maxi) - min(maxi, mini) + 1
            minlen = min (minlen, len)
        }
    }
    if (A[i] == min) {
        mini = i
        if ( maxi != -1) {
            len = max(mini, maxi) - min(maxi, mini) + 1
            minlen = min (minlen, len)
        }
    }
}
```

r > l

return minlen

$1 \leq N \leq 10^6$

$10^4 \leq N \leq 10^6$

5 second.