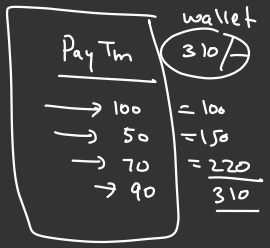


Happy Lohri / Makar Sakranti & Pongal

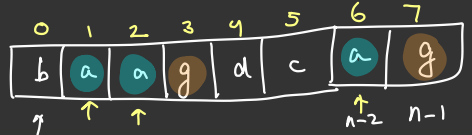
Carry Forward Techniques on Arrays

↳ carry forward
Some information (from the prev result)
↓
integer

Q. Given a character array (S) calculate no of pairs of (i, j) such that $i < j$ and

$S[i] == 'a'$ && $S[j] == 'g'$

All chars are in lowercase.



Pairs

<1,3>
<1,7>
<2,3>
<2,7>
<6,7>

5 Pairs

cnt = 0

Brute Force

```
for (i=0 , i<=n-2 , i++) {
    for (j=i+1 ; j<=n-1 , j++) {
        if (s[i]=='a' && s[j]=='g') {
            cnt = cnt + 1
        }
    }
}
print (cnt)
```

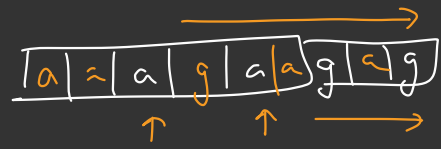
Time → $O(N^2)$

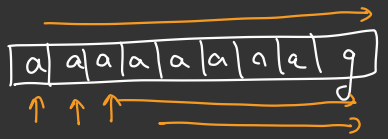
Space → $O(1)$

optimisation - Skip positions where s[i] != 'a'

```
cnt = 0
for (i = 0 , i <= n-2 , i++) {
    if ( s[i] == 'a' ) {
        for ( j = i+1 ; j <= n-1 , j++ ) {
            if ( s[j] == 'g' ) {
                cnt = cnt + 1
            }
        }
    }
}
print (cnt)
```

O(N^2) time
O(1) space



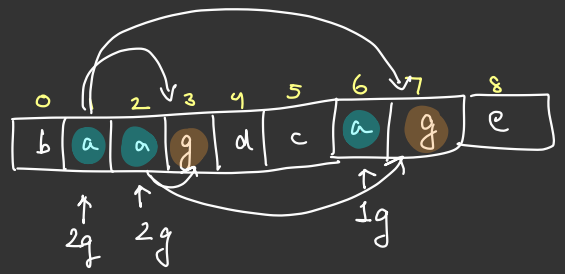


worst case

Count of g's after every a (Right)

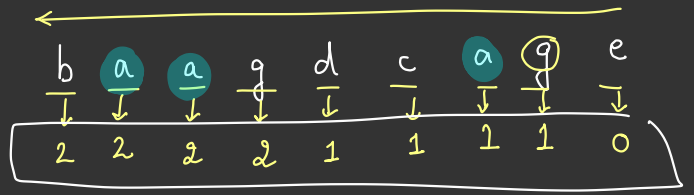
Another Attempt

S =



Time = O(N+N)
= O(N)

Space = O(N)



garr →

ans = 2 + 2
+ 1
= 5

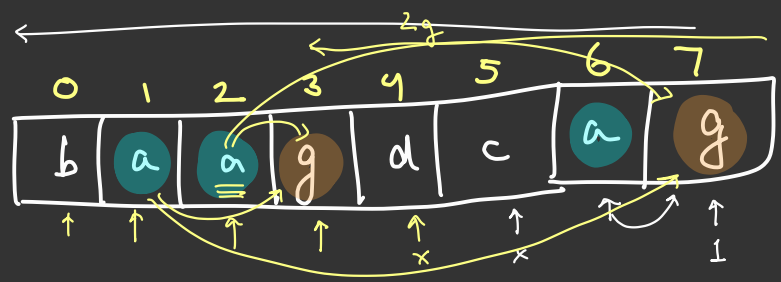
garr[i] = denotes no of g's having idx ≥ i

Independent
2 loops

→ Updating count of g
→ looking for an 'a'

] Combine in one loop

S =



ans = 0 + 1 + 2 + 2

g_cnt = 2

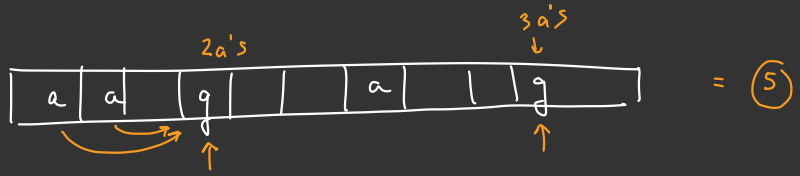
Time = O(N)

Space = O(1)

```
int g_cnt = 0 , ans = 0

for (i = n-1;  i >= 0 , i-- ){
        if ( s[i] == 'g' ){ g_cnt ++ }
        else if( s[i] == 'a' ) { ans = ans + g_cnt }
}

print (ans)
```



Right to Left

Carry forward count of g's for every a

Left to Right [DIY] - Hw

Carry forward count of a's before every g

Q2 Leaders in Array

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

An arr[i] will be leader if it is strictly greater than all elements on its Right Side

Note: last element a[n-1] is always counted as leader.

Example -

0	1	2	3	4	5	6	7	
15	-1	7	2	5	4	2	3	5

arr[i] is leader if arr[i] > max(a[i+1], ..., a[n-1])

longest after idx i

Brute Force → for (every i) {

// find out largest i+1 --- n-1

for (j = i+1; j <= n-1; j++) {

== Track Largest

}

⇒ Possibly optimise ?

if (arr[i] > largest) {

cnt = cnt + 1

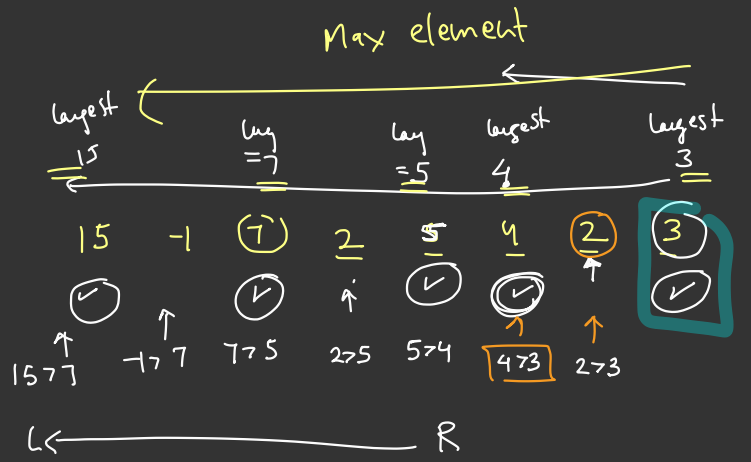
}

}

Time : $O(N^2)$

Space : $O(1)$

Can we iterate once from Right to left & Maintain Largest So Far



largest = ~~3~~ ~~4~~ ~~5~~ ~~7~~
15

Time. O(N)

Space. O(1)

```
largest = a[n-1]
count = 1
for (i = n-2; i>=0; i--){
    if (a[i] > largest){
        count++
        largest = a[i]
    }
}
print(count)
```

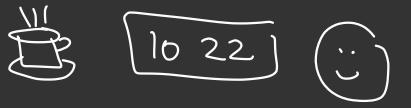
PS

← Extra array

Prefix Sum

↓

Range Queries [L,R] related to Sum



CF

→ count how many a's so far

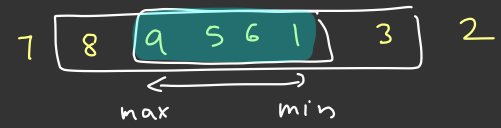
Tracking a single variable at every idx { → max so far

Challenge

Given N array elements, find out length of smallest subarray which contains both min & max of the array

Duplicates may be present

Examples

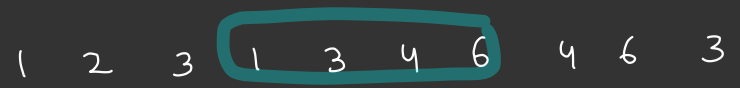


Min = 1

Max = 9

Output = 4

Ex =



Min = 1

Max = 6

Length = 4 output

Observations :

(1) Find out Min / Max

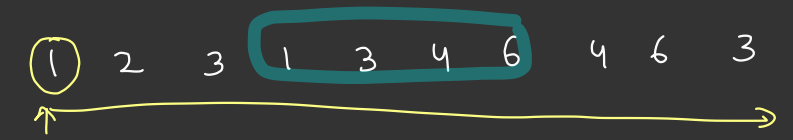
(2) Min - max will be extreme ends of subarray

Case - I

or

Case - II

Brute Force




```
Min = arr[0]
Max = arr[0]

for(i=1; i<=n-1; i++){
    Min = Math.min(a[i], Min); → O(1)
    Max = Math.max(a[i], Max); → O(1)
}
```

O(N)

Min = 1
Max = 6

⇒

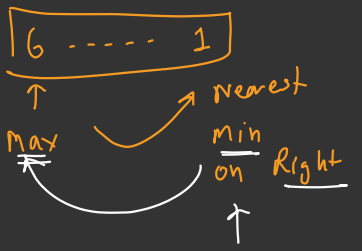
8 - 3 + 1
= 6

3, 4, 5, 6, 7, 8

```
for(i=0; i<=n-1; i++){
```

Case - I

- if you encounter Min at arr[i]
  - ↳ look for nearest max in range (i+1, n-1) } loop (J)
  - ↳ update window len if new len is smaller
- if you encntr a Max at arr[i] } loop (J)



Case-II

{

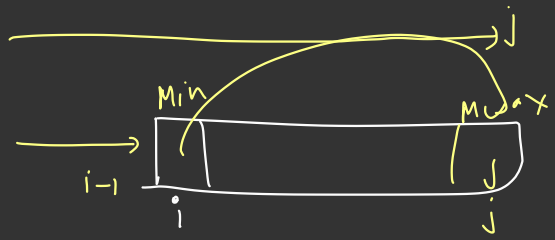
}

→ ↳ look for nearest min

↳ Update window len (i+1, n-1) if new len smaller

TC : $O(N^2)$

SC : $O(1)$

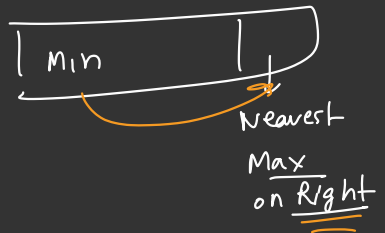


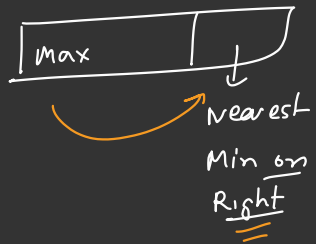
newlen = j - i + 1

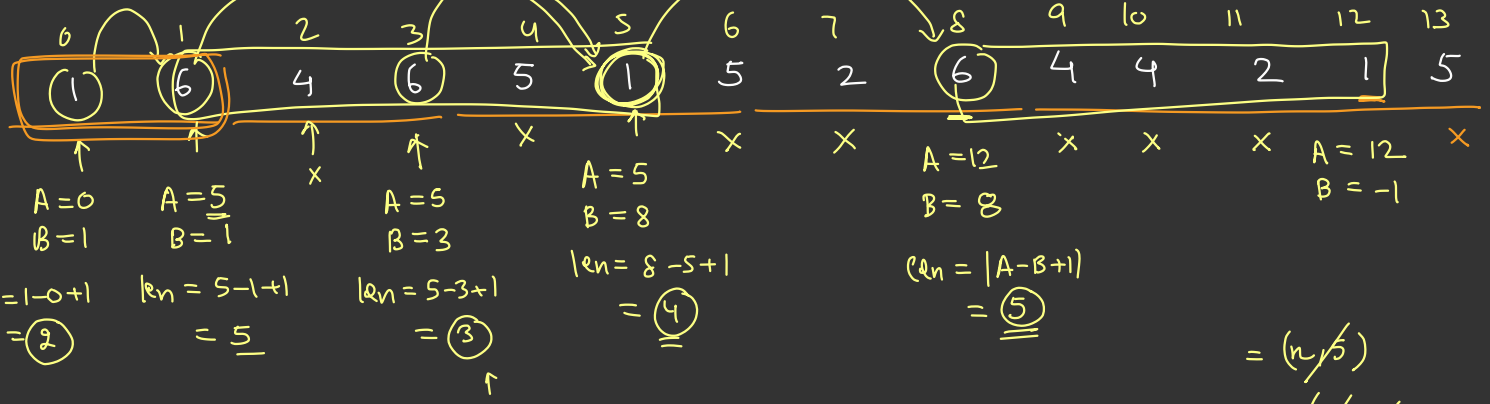
j - (i-1)

= j - i + 1

for every element which is min or max







Min = 1
Max = 6

A, B
↑
Carry Forward
R to L

A ⇒ min Idx
B ⇒ max Idx

= (n, 5)
ans = 5 4 3 2
= (5, 4)
= (4, 3)
= (3, 2)
= 3

(I)

```
Min = arr[0]
Max = arr[0]
for(i=1; i<=n-1; i++){
    Min = Math.min(a[i], Min); → O(1)
    Max = Math.max(a[i], Max); → O(1)
}
```

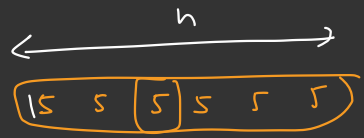
(II)

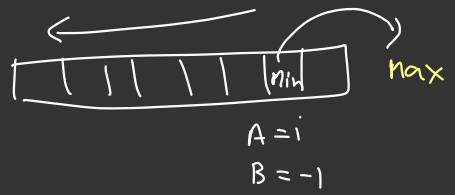
```
if(Min == Max)
    return 1
```
} Spl case

```
A = -1, B = -1, ans = n
for(i=n-1; i>=0, i--){
    if(arr[i] == min){
        A = i
        if(B != -1){ len = B-A+1 }
        ans = min(ans, len)
    }
}
```

O(N) TC

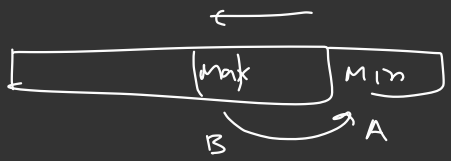
O(1) SC





Max → Min

```
else if (arr[i] == max) {
    B = i
    if (A != -1) {
        len = A - B + 1
        ans = min(ans, len)
    }
}
print(ans)
```



Given an array of integers A, find and return the product array of the same size where the ith element of the product array will be equal to the product of all the elements divided by the ith element of the array. Note: It is always possible to form the product array with integer (32 bit) values. Solve it without using the division operator.Input 1:
A = [1, 2, 3, 4, 5]
Output 1:
[120, 60, 40, 30, 24]