Subsequences 'ag'

Given a character array.

Count the no. of pairs (i, j) s.t. $i < j$ & $A[i] = 'a'$, $A[j] = 'g'$.

All characters are lowercase alphabets.

$A = [\underset{0}{b}\ \underset{1}{a}\ \underset{2}{a}\ \underset{3}{g}\ \underset{4}{d}\ \underset{5}{c}\ \underset{6}{a}\ \underset{7}{g}]$

(1, 3) (2, 3) (6, 7) (1, 7) (2, 7)

Ans = 5.

$A = [\underset{0}{g}\ \underset{1}{c}\ \underset{2}{a}\ \underset{3}{g}\ \underset{4}{g}\ \underset{5}{a}\ \underset{6}{a}\ \underset{7}{g}]$

(2, 3) (2, 4) (2, 7) (5, 7) (6, 7)

Ans = 5

Soln ▷ Brute force

~~∀ i, j where i < j~~ ⇒ Iterate & check

if (A[i] == 'a' & A[j] == 'g')

(for j = i+1 ...)

⇓ optimise

Code

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

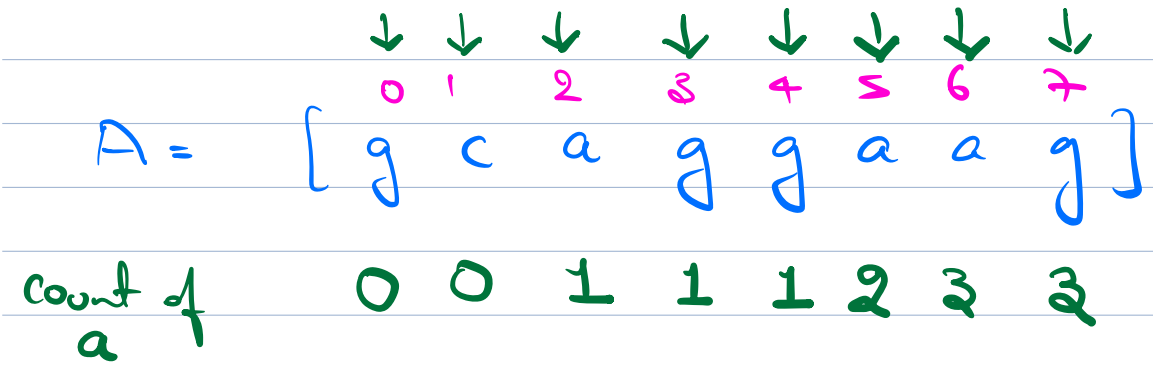
T.C. = $O(N^2)$

2) Optimised Soln

⇒ ∀ A[i] where A[i] = 'a', you need the count of 'g's in the right side

OR

⇒ ∀ A[i] where A[i] = 'g', you need the count of 'a's in the left side



index	ans
2	+1
4	+1
7	+3

> (5)

Carry forward

→ If the answer of i, can be calculated optimally using the answer of i-1.

We carry forward the ans of (i-1) to i

Code

```
ans = 0;
count = 0;
for (i=0; i<N; i++) {
        if (A[i] == 'a') {
            count++;
        }
        else if (A[i] == 'g') {
            ans += count;
        }
}
return ans;
```

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

Given an integer array A.
Count the no. of leaders in the array.

leader ⇒ An element which is greater than all the elements in the right

Note ⇒ A[N-1] is always a leader.

A: [15, 1, 7, 9, 5, 4, 2, 3]
0 1 2 3 4 5 6 7
✓ ✗ ✓ ✗ ✓ ✓ ✗ ✓

Ans = 5

A = [10, 7, 9, 3, 2, 4]
0 1 2 3 4 5
✓ ✗ ✓ ✗ ✗ ✓

Ans = 3

max in the right.

Soln

1) Brute force

⇒ ∀i, check if it is greater than all elements in the right.

Code

```
count = 1;  ⇒  // A[N-1];
for (i = 0; i < (N-1); i++) {

    bool isans = True;
```

```
for (j = i+1; j < N; j++) {
        if (A[j] >= A[i]) {
            isans = false;
            break;
        }
    }
    if (isans == True) {
        count ++;
    }
}
return count;
```

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

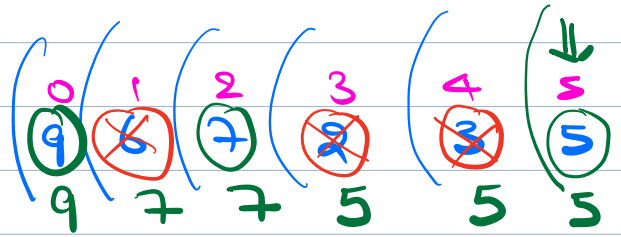
S.C. = $O(1)$

2) Optimised Sol^n

$\forall i \Rightarrow$ max element from $[i+1, N-1]$

$\Downarrow$

max in the right.



Ans = 3

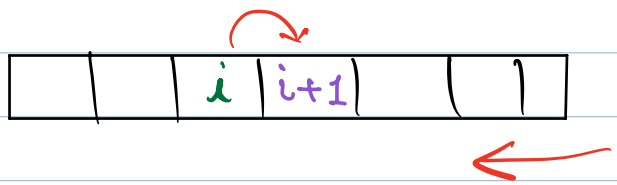
Code

```
ans = 1;
rmax = A[N-1];

for (i = N-2; i >= 0; i--) {

    if (A[i] > rmax) {
        ans++;
        rmax = A[i];
    }
}
return ans;
```

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



Subarray ⇒ Continuous part of array.

s e [s<=e]

A = [15, 1, 7, 2, 5, 4, 2, 3]
(0) (1) (2) (3) (4) (5) (6) (7)

[15, 1, 7, 2] ✓ ⇒ [0, 3]

[2] ✓

[5, 2, 7] ✗ (order should be same as order in A)

[s, e] ⇒ e - s + 1

Given an integer array of size N.

Find the length of the smallest subarray which contains both max & min of the array.

A: [2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1]
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

Find max & min ⇒ O(N)
(6) (1)

A: [8, 8, 8, 8, 8, 8, 8]
(0) (1) (2) (3) (4) (5) (6)

max = 8
min = 8

Ans = 1

Sol^

Obs 1 : Subarray will always start & end with max & min.

Obs 2 : Any subarray there will only be one max & one min

1) Brute force

∀ subarrays, check if start & end is (max, min) or (min, max)

↳ length = (e - s + 1)

↓

global_min.

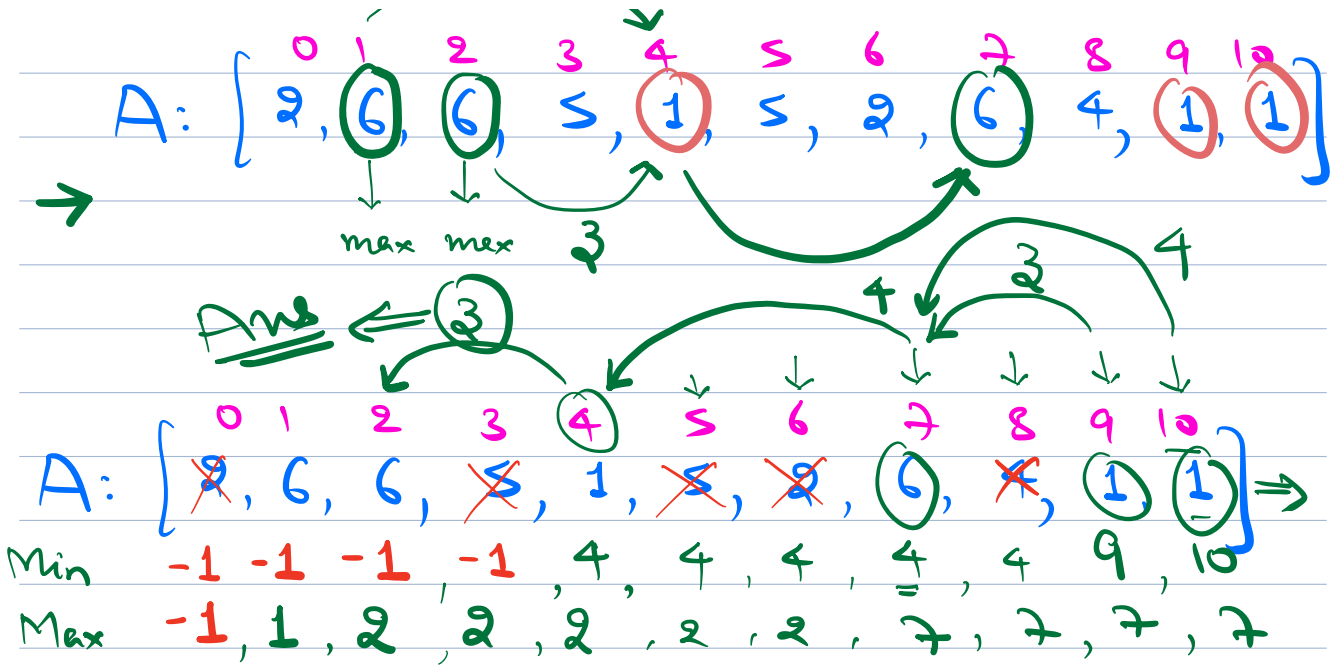
Code

```
lengthMin = N;
for (s = 0; s < N; s++) {
    for (e = s; e < N; e++) {
        // A[s, e] ⇒ subarray.
        if ((A[s] == min && A[e] == max) ||
            (A[s] == max && A[e] == min)) {
            l = e - s + 1;
            lengthMin = min (lengthMin, l);
        }
    }
}
```

T.C. = $O(N^2)$
S.C. = $O(1)$

---

2) Optimised Soln



⇒ for every min, I need nearest max in left

⇒ for every max, I need nearest min in left

Code

```
minInd = -1;
maxInd = -1;
ans = N;
for (i = 0; i < N; i++) {
    if (A[i] == max) {
        if (minInd != -1) {
```

```
                l = i - minInd + 1
                ans = min (ans, l);
            }
            maxInd = i;
        }
        else if (A[i] == min) {
            if (maxInd != -1) {
                l = i - maxInd + 1;
                ans = min (ans, l);
            }
            minInd = i;
        }
    }
    return ans;
```

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

★ Always focus on current
Do not live in past.

Doubt

Total Sum = 28

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

Ans

Lsum = 0 + 1 = 1 + 2 = 3 + 3 = 6 + 4 = 10

Rsum = 28 - 2 = 26 - 3 = 23 - 4 = 19 - 8 = 10