Q1 Given a char[], calculate no of pairs i, j such that $i < j$ and $s[i] = 'a'$ && $s[j] = 'g'$. All characters are lower-case. ['a' - 'z']

Ex1

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

$\Rightarrow$ <1,3>, <1,7>, <2,3>, <2,7>, <6,7>

ans = 5

Ex2

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

<1,3>, <1,4>, <1,7>, <5,7>, <6,7>

ans = 5

Ex3

0	1	2	3	4	5	6
a	c	g	d	g	a	g

<0,2>, <0,4>, <0,6>, <5,6>

ans = 4

```
//Check all the pairs
c = 0
for(i = 0; i < N; i++){
    for(j = i+1; j < N; j++){
        if(s[i] == 'a' && s[j] == 'g'){
            c++
        }
    }
}
return c;
```

TC : $O(N^2)$
SC : $O(1)$

```
//Check all the pairs
c = 0
for(i = 0; i < N; i++){
    if(s[i] == 'a'){
        for(j = i+1; j < N; j++){
            if(s[j] == 'g'){
                c++
            }
        }
    }
}
return c;
```

TC : $O(N^2)$
SC : $O(1)$

if s[i] == 'a' :
(We are counting no of 'g's on the right side)

Ex 4

0	1	2	3	4	5	6	7	8
a	d	g	a	g	a	g	f	g

i = 0 → cnt = 4

i = 3 → cnt = 3

i = 5 → cnt = 2

ans = 9

Right to left.

Ex 4

0	1	2	3	4	5	6	7	8
a	d	g	a	g	a	g	f	g
	x	c=c+1	ans=ans+c	c=c+1	ans=ans+c	c=c+1	*	c=c+1
ans=ans+c		c=4	ans=5	c=3	ans=2	c=2		c=1
ans=9								

ans=0
c=0

//Pseudo code

ans=0, c=0

⇒for(i=N-1; i>=0; i--){

if(arr[i]=='g'){

c=c+1;

}else if(arr[i]=='a'){

ans=ans+c;

}

}

return ans;

TC : O(N)

SC : O(1)

Idea 2

For every 'g' calculate no' of a on left side.

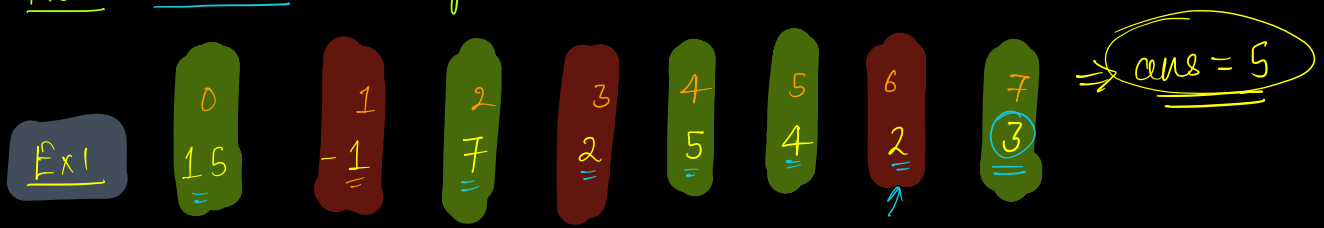
1) ans=0, c=0
2) if a, c++
3) if g, ans=ans+c
4) Left to Right (i=0; i<N; i++)

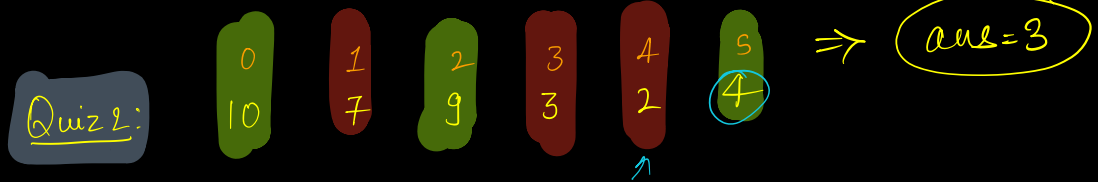
Q2 Given an array, you have to find all leaders in array.

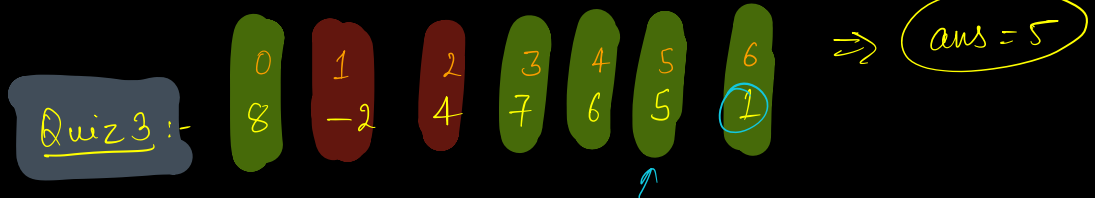
An element is a leader, if it is strictly greater than all elements on its right side.

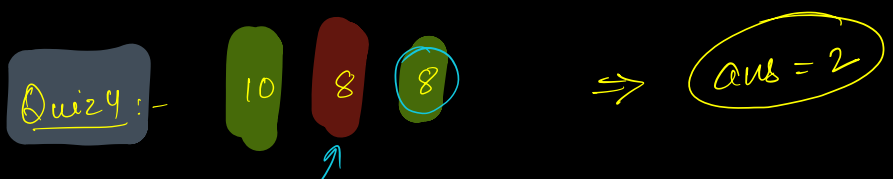
→ strictly greater than max on the right.

Note:- arr[N-1] is always considered a leader









```
//Right to left.
//Carry max val.
ans = 1
max = ar[N-1]
for(i = N-2; i >= 0; i--){
    //ar[i] is a leader or not
    if(ar[i] > max){
        ans = ans + 1
        max = ar[i];
    }
}
return ans;
```

max contains max of all elements on right side.

TC : O(N)
SC : O(1)

Subarray Basics:

1) Continuous part of an array is called as Subarray.

(i) A single element is a subarray.

(ii) Entire array is a subarray.

(iii) Empty cannot be a subarray.

	0	1	2	3	4	5	6	7	8
arr[] →	-3	4	6	2	8	7	14	9	21

Ex1 indices [2, 3, 4, 5] → [2 - 5]

Ex2 indices [3 4 6 7 8] → } missing index 5

Ex3 indices [1 2 3] → [1 - 3]

Ex4 indices [5] → [5 - 5]

For example, Subarray [3 - 7] ⇒ [3, 4, 5, 6, 7] = 5

↳ length : → 7 - 3 + 1 = 5

↳ Subarray [start, end] : length : → end - start + 1

Usage of Predefined function

- max(a,b) → TC : O(1) → returns max of 2 Nos.
- min(a,b) → TC : O(1) → returns min of 2 Nos.
- sort( ) → TC : O(NlogN) → Sorts N Numbers.

||Note : Before using any predefined fun, read its TC.

```
// Given an array, find min_val.
min_val = arr[0]; // ∞
for(i = 0; i < N; i++){
    if (arr[i] < min_val) {
        min_val = arr[i];
    }
}
```

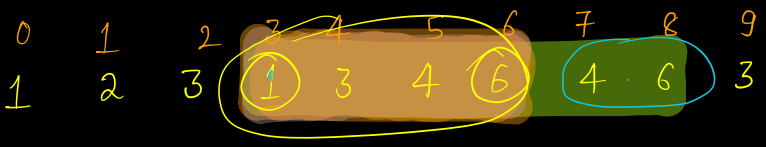
min_val = min(min_val, arr[i]);

TC : O(N)
SC : O(1)

Q3) Given an array, find the length of smallest subarray which contains both min & max of array.

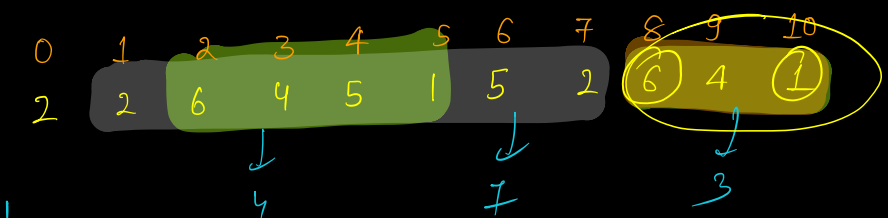
min_val = 1, max_val = 6

Ex1



ind: [3, 8], len = 6
ind: [3, 6], len = 4 } ans = 4

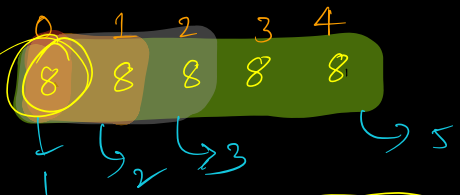
Quiz 1



min = 1
max = 6

ind: [1 . 7] : len = 7
ind: [2 5] : len = 4
ind: [8 10] : len = 3 } ans = 3

Quiz 3



min = 8
max = 8

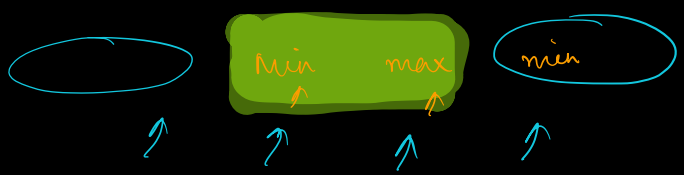
ans = 1

```
if(min == max)
  return 1;
```

#Observations

Final ans subarray

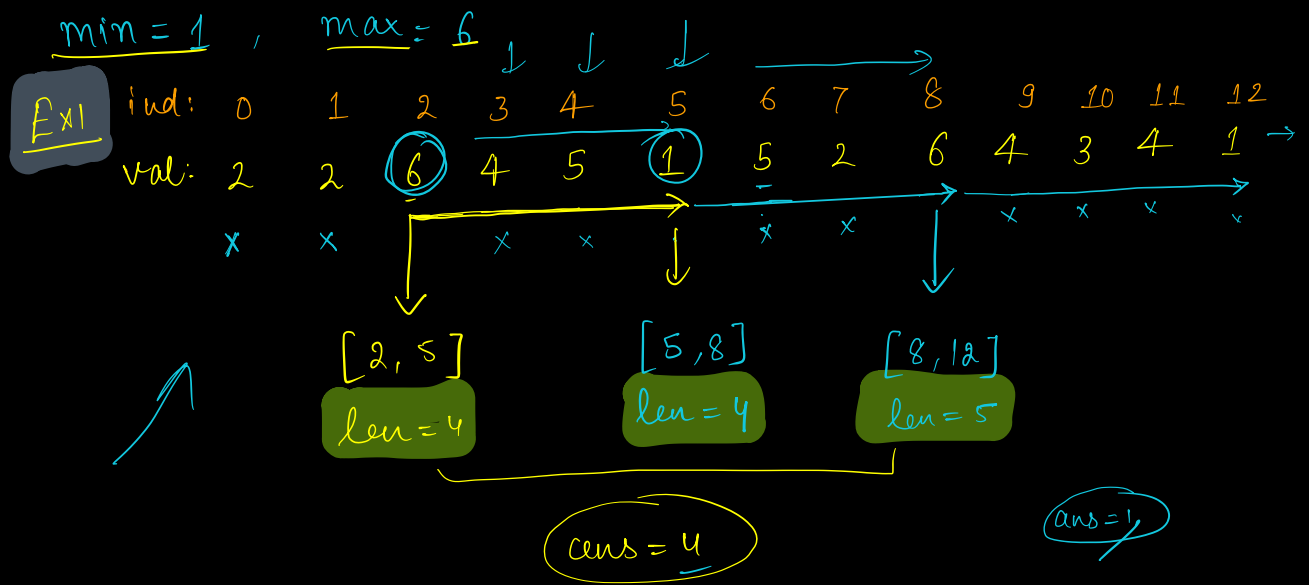
① It should contain 1 min & 1 max.

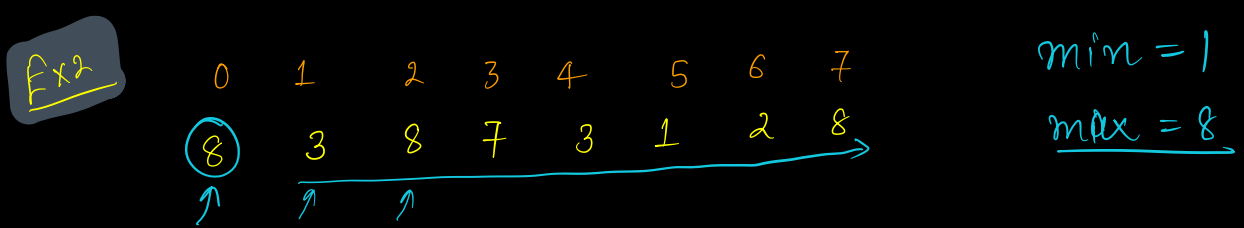


② Min & Max should only be on corners.

③ Case 1: [Min Max] ⇒ If you are at min, you need nearest max on right.

Case 2: [Max Min] ⇒ If you are at max, you need nearest min on right


min = 1 , max = 6
Ex1
ind: 0 1 2 3 4 5 6 7 8 9 10 11 12
val: 2 2 6 4 5 1 5 2 6 4 3 4 1
[2, 5]
len = 4
[5, 8]
len = 4
[8, 12]
len = 5
cans = 4
ans = 1


Ex2
0 1 2 3 4 5 6 7
8 3 8 7 3 1 2 8
min = 1
max = 8

// Pseudo code.

iterate & get min_val of array // min_val
iterate & get max_val of array // max_val.

```
if(min_val == max_val)
    return 1;

ans = N
for(i = 0; i < N; i++){
    if(arr[i] == min_val){ // Nearest max on right side.
        for(j = i+1; j < N; j++){
            if(arr[j] == min_val) break; // lies
            if(arr[j] == max_val){
                ans = min(ans, j-i+1);
                break;
            }
        }
    }
    else if(arr[i] == max_val){
        for(j = i+1; j < N; j++){
            if(arr[j] == max_val) break;
            if(arr[j] == min_val){
                ans = min(ans, j-i+1);
                break;
            }
        }
    }
return ans;
```

TODO :-

O(N)

TC : O(N2)
SC : O(1)

nearest

min_i // store min value's index on right side

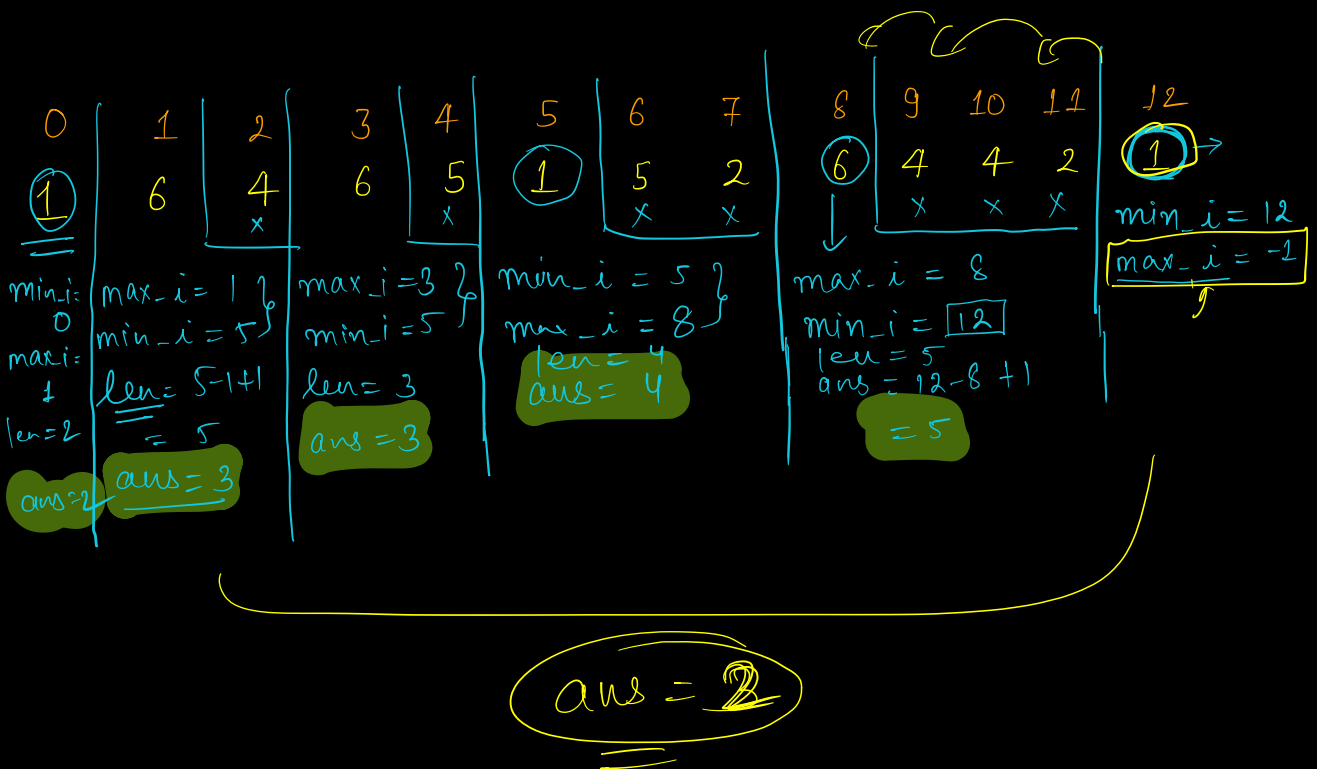
max_i // store max value's nearest index on right side.

Ex:

max = 6

min = 1

{ min_i = -1
{ max_i = -1
{ ans = N



ans = 2

Pseudo code

N = length of the subarray.

Iterate & get min_val & max_val.

if (min_val == max_val)

return 1;

ans = N, min_i = -1, max_i = -1

max ← 1 → min

for(i = N-1; i >= 0; i--) {

if (arr[i] == min_val) {

min_i = i;

if(max_i != -1) { len = max_i - min_i + 1; ans = min(ans, len); } [min_i ... max_i]

there is max on right

} else if (arr[i] == max_val) {

max_i = i;

if(min_i != -1) { len = min_i - max_i + 1; ans = min(ans, len); } [max_i ... min_i]

there is min on right

}

return ans;

TC : O(N)

SC : O(1)

Doubts

→ Subarrays

→ 2D Matrices

→ Interview Problems on Arrays

} 6 sessions

Bits Manipulation..