CF & Subarrays

Good Morning Everyone :)

Content

Count pairs "ag"

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

Note :- All characters are in lower case.

Constraints : $1 \le N \le 10^5$

'a' ≤ s[i] ≤ 'z'

s[8] = { b a a g d c a g }

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

Pairs → (1, 3) & (1, 7)

(2, 3) && (2, 7)

(6, 7)

\# ans = 5

s[8] = { b c a g g a a g }

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

Pairs = (2, 3), (2, 4) & (2, 7)

(5, 7)

(6, 7)

\# ans = 5

s[7] = { a c g d g a g }
0 1 2 3 4 5 6

Pairs = (0,2), (0,4) & (0,6)

(5,6)

ans = 4

Brute Force → Check for all the valid pairs if

s[i] == 'a' && s[j] == 'g' → increase my count

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

TC: O(n²)

SC: O(1)

Idea 2 → Iterate on the right hand side only when you have the first character as 'a'

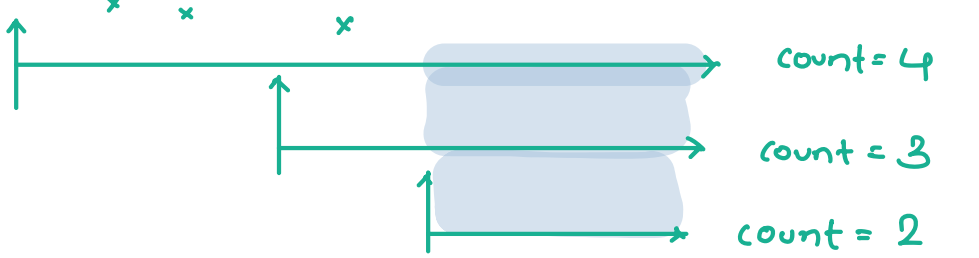
```
count = 0
for (i=0; i<n; i++)
    if (s[i] == 'a') {
        for (j = i+1; j<n; j++) {
            if (s[j] == 'g') count ++;
        }
    }
return count;
```

TC: $O(N^2)$

SC: $O(1)$

Idea 3

Eg: ar[] = { a d g a g a g f g }



count = 9

Optimal Approach → Count the no. of 'g' from RHS & update your answer as soon as we find 'a'

ans=0 countg = 0

a	d	g	a	g	a	g	f	g
ans=5 ans = ans+c ans=9		c=3 c=c+1 c=4	ans=2 ans = ans+c ans=5	c=2 c=c+1 c=3	ans = ans+c = 2	c=1 c=c+1 c=2		c=0 c=c+1 c=1

Ans = 9

ans=0 countg = 0

a	g	a	g	g
ans=2 ans+=c ans=5	c=2 c=c+1 c=3	ans=0 ans+=c ans=2	c=1 c=c+1 c=2	c=0 c=c+1 c=1

```
ans = 0   c = 0
for ( i = n-1 ; i ≥ 0 ; i--) {
    if ( s[i] == 'g' ) c = c+1
    if ( s[i] == 'a' ) ans = ans + c
}
return ans;
```

TC : O(n)
SC : O(1)

RHS ⟶ LHS ⇒ count of 'g's for all a's

LHS ⟶ RHS ⇒ count of 'a' for particular 'g's

* Introduction of Subarray

Subarray → Continous part of the array
→ Single element is also a subarray
→ Entire array is also a subarray

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

[2 5] = 2, 3, -1, 6 ⟶ Yes, length = 4

2, 9, 8 ⟶ No

2, 1, 4 ⟶ No

8 ⟶ Yes, length = 1

Quiz

A[] = { 2, 4, 1, 6, -3, 7, 8, 4 }
0 1 2 3 4 5 6 7

{1, 6, 8}

{1, 4}

{6, 1, 4, 2}

{7, 8, 4}

A[] = { 4, 5, 1, 9, 0, 2, 3, 5 }

(a) [5]

(b) [4, 5, 1, 0]

(c) [9, 0, 2, 3]

(d) [4, 5, 1]

* Properties of Subarray

01. Represent a subarray [si ei]

02 length of subarray ⇒ ei - si + 1

03. A[ ] = { 4, 2, 10, 3 }
0 1 2 3

st=0	st=1	st=2	st=3
[0 0]	[1,1]	[2,2]	[3,3]
[0 1]	[1,2]	[2,3]	
[0 2]	[1,3]		
[0 3]			
4	+ 3	+ 2	+ 1

Total no. of subarr = 4+3+2+1

$= \frac{n*(n+1)}{2}$

n = length of array

Q Given an arr[ ], Print the subarray from s to e

A[ ] = { 4, 1, 2, 3, 5, 4 } s=2 e=5
0 1 2 3 4 5

Idea → Print the ele from s to e using a loop

```
for (i = s ; i ≤ e ; i++){
    print(A[i]);
}
```

TC: O(n)
SC: O(1)

Q Given an A[N], print all subarrays

Note :- Print every subarray in a new line

A[ ] = { 2, 8, -1, 4 }
0 1 2 3

st=0 [0, 0] = {2}
[0, 1] = {2, 8}
[0, 2] = {2, 8, -1}
[0, 3] = {2, 8, -1, 4}

st=1 [1, 1] = {8}
[1, 2] = {8, -1}
[1, 3] = {8, -1, 4}

st=2 [2 2] = {-1}
[2 3] = {-1, 4}

st=3 [3 3] = {4}

i	j	i	j	i	j	i	j
[0,0]		[1,1]		[2,2]		[3,3]	
[0,1]		[1,2]		[2,3]			
[0,2]		[1,3]					
[0,3]							

Idea → To print a subarray, we need s & e

```
for (i=0; i<n; i++){                    st=?
    for(j=i; j<n; j++){                 end=j
        st=i , end=j
        for(k=st; k<=end; k++){
            print(A[k])
        }
        println();
    }
}
```

TC: $O(n^3)$

SC: $O(1)$

Q Given an ar[N], return the smallest subarr which contains both max & min element of array.

A[] = {	1	2	3	1	3	4	6	4	6	3 }
	0	1	2	3	4	5	6	7	8	9

min = 1

max = 6

sub[0 6] = 6-0+1 = 7

sub[3 8] = 8-3+1 = 6

sub[3 6] = 6-3+1 = 4

ans = 4

A[] = { 2 2 6 4 5 1 5 2 6 4 1 }
0 1 2 3 4 5 6 7 8 9 10

min = 1
max = 6

Ans = 3

A[] = { 8, 8, 8, 8 }
0 1 2 3

min = 8
max = 8

Ans = 1

Observation

01. If min & max are same, ans = 1

02 How many min & max do we want in our subarray?

1, 6, 1, 6

1, 2, 5, 1, 6

1 2 1 3 4

1 2 5 5 6 6

for smallest subarr, we should only have 1 min & 1 max

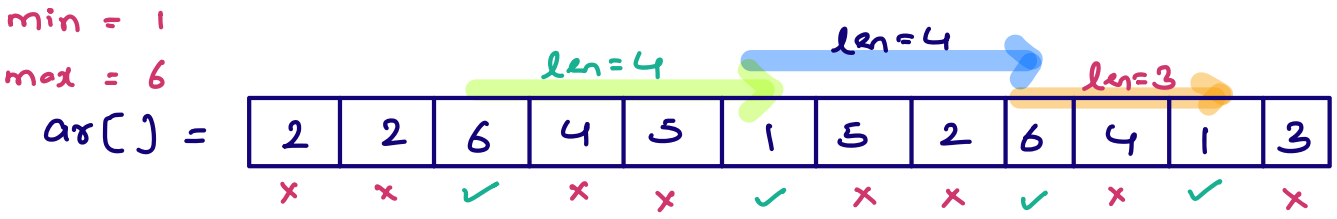
03 Position of min & max? → should be at corners

A[] = { .... min ... max .... }

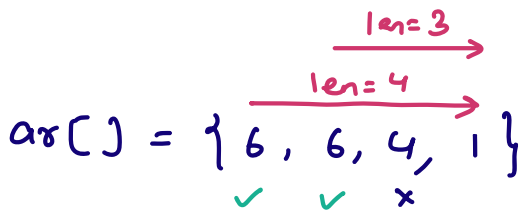
A[] = { ... max ... min ... }

Brute force → if (A[i] == min) → need to look for closest max on RHS

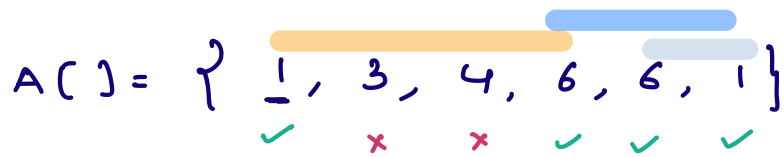
if (A[i] == max) → need to look for closest min on RHS



Ans = 3



Ans = 3



len = ∞ vs 4 = 4

len = 4 vs 3 = 3

len = 3 vs 2 = 2

Ans = 2

* Iterate & find min & max of array

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

ans = n // max length of subarr having min & max at corners

for (i=0; i<n; i++){

    if (A[i] == min){
```

```
for (j = i+1; j<n; j++){
    if (A[j] == max){
        ans = Math.min (ans, j - i + 1);
        break;
    }
}

if (A[i] == max){          → look for min
    for (j = i+1; j<n; j++){
        if (A[j] == min){
            ans = Math.min (ans, j - i + 1);
            break;
        }
    }
}
```

TC: $O(n^2)$

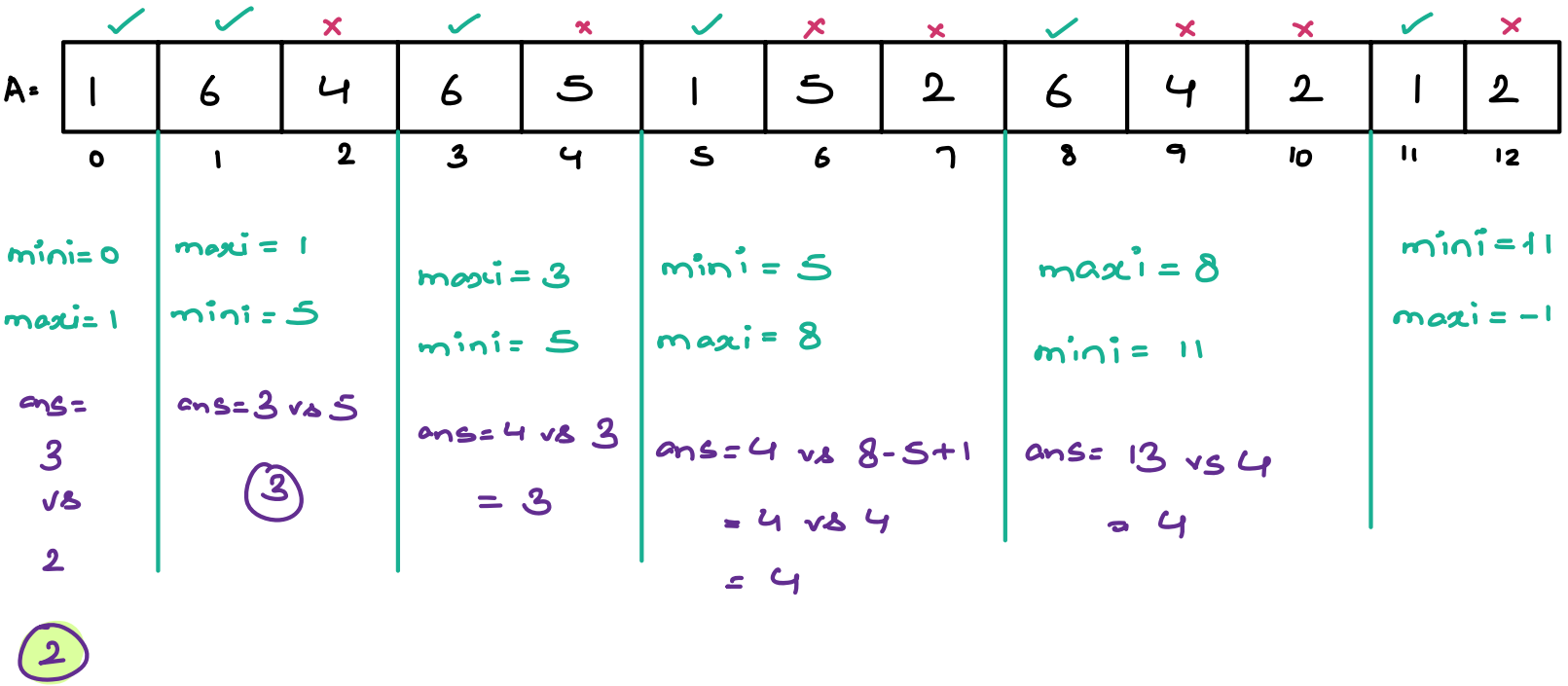
SC: $O(1)$

Optimal Idea → Carry forward the mini & maxi

A:	1	6	4	6	5	1	5	2	6	4	2	1	2
	0	1	2	3	4	5	6	7	8	9	10	11	12

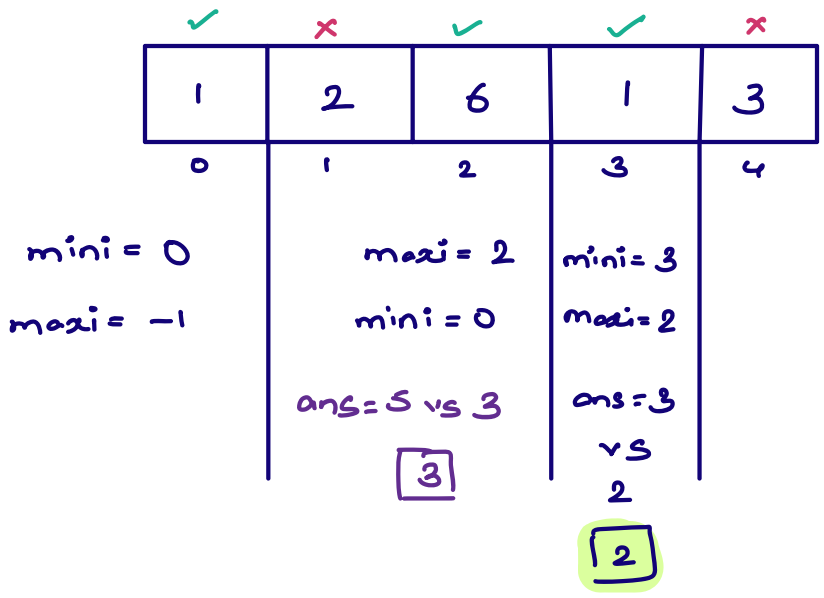
min = 1 mini = -1 ans = n

max = 6 maxi = -1



2

LHS →



min=1
max=6
mini=-1
maxi=-1
ans=5

Ans = 2

* Iterate & find min & max

```
if (min == max) return 1;
ans = n;
maxi = -1 ;
mini = -1;
```

```
for (i=n-1; i≥0; i--){
    if (A[i]==min){
        mini=i;
        if (maxi!=-1){
            ans= Math.min(ans, maxi-mini+1) :
        }
    }
    if (A[i]==max){
        maxi = i
        if (mini!=-1){
            ans= Math.min(ans, mini-maxi+1);
        }
    }
}
}
```

TC : O(n)
SC : O(1)

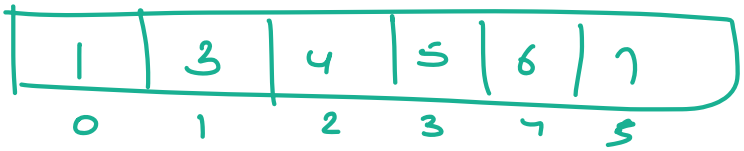
Doubts

Prefix Array Carryforward

range sum queries

↓

Subarray / continous part of array



Pf :- [1, 4, 8, 103, 19, ...]

[0 2]

[3 4]

[4 5]

} Cf approach

[0 3]