Today's Content

AC pairs

Closest min max

Subarray definition

Leaders in an array

Ques) Count Pairs "ag"

Given a char[], calculate no. of pairs i,j s.t, i<j, s[i]=='a' && s[j]=='g'.

Note:- All characters are lower case.

e.g.)

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

pairs :- <1,3>, <2,3>, <1,7>, <2,7>, <6,7> => 5 pairs.

e.g 2)

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

<2,3>, <2,4>, <2,7>, <5,7>, <6,7>

=> 5 pairs

e.g.3)

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

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

=> 4 pairs.

idea 1) a) Check all pairs (i, j)

T.C → O(n²)
S.C → O(1).

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

idea 2)

T.C → O(n²)
S.C → O(1)

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

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

cnt += 4

cnt += 3

cnt += 2

9

C → how many g's are to the right,

Ans = 0 c = 0

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

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

T.C → O(N)
S.C → O(1)

T.C → O(N)
S.C → O(1)

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

ans += 1

ans += 2

ans += 3

ans += 3

20) Leaders in an Array.

Last element is always leader.

Given an Ar[n], you have to count leaders in arr[].

An ele is leader if it is strictly greater than max of elements in its right.

Ex1: 15, -1, 7, 2, 5, 4, 2, 3 (indices 0 1 2 3 4 5 6 7)

Ans = 5.

Ex2: 10, 7, 9, 3, 2, 4 (indices 0 1 2 3 4 5) → 3

Ex3: arr[7] = 8, -2, 4, 7, 6, 5, 1 (indices 0 1 2 3 4 5 6) → 5

Ex4:- 10, 8, 8 (indices 0 1 2) → 2.

:- Bf :- for every element, iterate to right & get max.

T.C → O(N²)
S.C → O(1)

```
for (i=0; i<N; i++) {
    // iterate & get max to right
    for (j=i+1; j<N; j++)
}
```

ans = 5
max = 15

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

```
ans = 1
max = arr[N-1];
for (i = N-2; i>=0; i--) {
    if (arr[i] > max) {
        ans++;
        max = arr[i]
    }
}
return ans;
```

T.C → O(N)
S.C → O(1)

8 8 8

ans = 1
max = 8

Subarray Basics

1) Continuous part of an array is called subarray.

→ Single element is a subarray.
→ full array is also a subarray.

	0	1	2	3	4	5	6	7	8
Ex:-	-3	4	6	2	8	7	14	9	21

indices [2, 3, 4, 5]

indices [3, 4, 6, 7, 8]

→ [5]

s e → e-s+1

ind → [2, 8] → 8-2+1 = 7

ind [s e] → e-s+1

	0	1	2	3	4	5	6	7	8
Ex:-	-3	4	6	2	8	7	14	9	21

Subset / subsequence

↳ need not to be continuous.

indices

↳ {2, 4} → {0, 5, 8}

↳ {7, 2} → {3}

↳ this is also a subset

Break : 9:54 - 10:04 pm

Ques) Closest Min max :-

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

Ex 1)

0	1	2	3	4	5	6	7	8	9
1	2	3	1	3	4	6	4	6	3

min = 1

max = 6

ans = 4.

Ex 2)

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	1

Observations :- 1) we need only 1 min & 1 max.

Subarray min min max

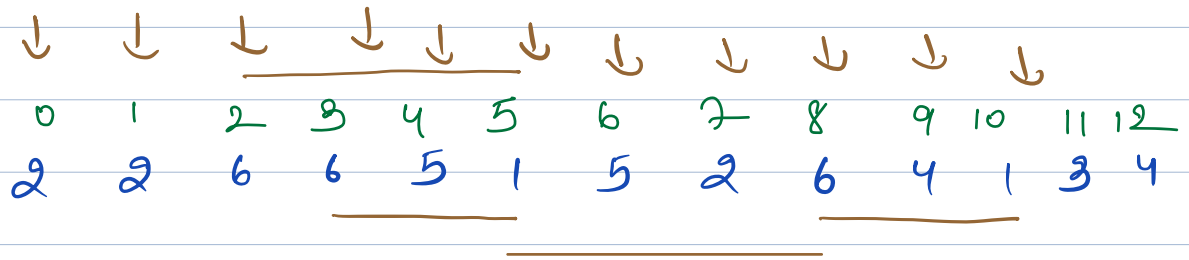
2) Min & max elements will always be in corner.

3] [min — max] [max — min]

Ex 8 8 8 → 1

min = 8, max = 8

(max — min) (min — max)



max = 6

min = 1

```
// iterate & get min & max.
if (min == max) {
    return 1
}
ans = n;
for (i=0; i<n; i++) {
    if (arr[i] == min) {
        for (j=i+1; j<n; j++) {
            if (arr[j] == max) {
                ans = min(ans, j-i+1);
                break;
            }
        }
    }
    if (arr[i] == max) {
        for (j=i+1; j<n; j++) {
            if (arr[j] == min) {
                ans = min(ans, j-i+1);
```

T.C -> O(n^2)

S.C -> O(1)

// optimize

min = 1
maxe = 6

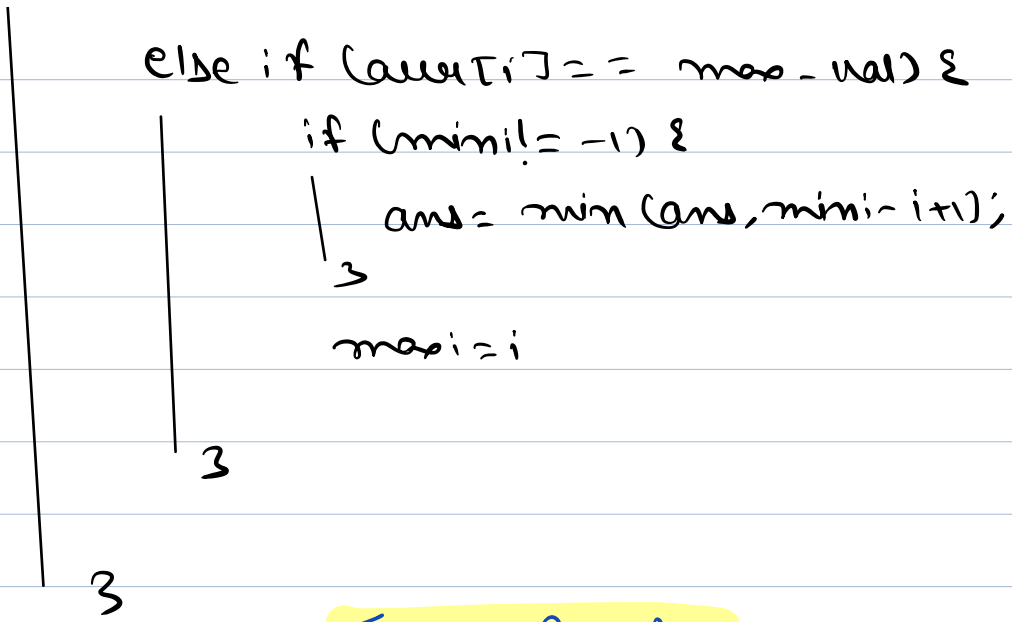
mini = ~~1~~ ~~5~~ 0
maxi = ~~8~~ ~~3~~ 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	6	4	6	5	1	5	2	6	4	4	2	1	5
*	*	*	*	*	*	*	*	*	*	*	*	*	*

Ans: ~~5~~ ~~4~~ ~~3~~ 2

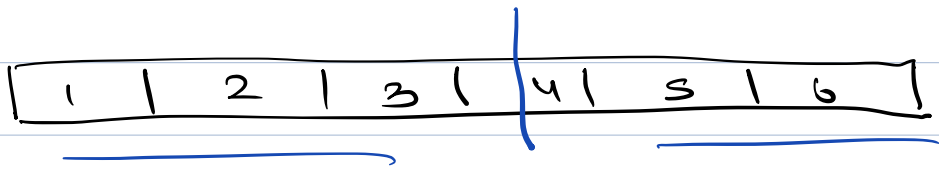
1) Iterate & get min_val & max_val.
2) if (minval == maxval) { return 1 }
   ans = N, maxi = -1, mini = -1.

```
for (i = N-1; i >= 0; i--) {
    if (arr[i] == min_val) {
        if (maxi != -1) {
            ans = min(ans, maxi - i + 1);
        }
        mini = i
    }
```



T.C → O(N)

S.C → O(1).



Prefix product

Suffix product.

2 3 4

5 6 7

8 9 10