SCALER

Array - Carry forward & Subarrays

TABLE OF CONTENTS

Count 'a-g' pairs

< *Question* > : Given a string s of lowercase characters, return the count of pairs (i, j) such that i < j and s[i] is 'a' and s[j] is 'g'.

a-g a is before g

0,3 0,5 4,5

a b e g a g

0 1 2 3 4 5

ans = 3

0,2 0,4 0,6

a c g d g a g

0 1 2 3 4 5 6

5,6 ans = 4

2,3 2,4 2,7

str → b c a g g a a g

0 1 2 3 4 5 6 7

5,7

ans = 5 6,7

BF Idea I can check all possible pairs

< / > Code

```
count = 0
for (i: 0 → N-1) {
    for (j: 0 → N-1) {
        if (i<j && s[i]=='a' && s[j]=='g')
            count++
    }
}
```

TC: $O(N^2)$

SC: $O(1)$

Idea Carry Forward

g will create pairs with all 'a' on the left

str →	b	c	a	g	g	a	a	g
	0	1	2	3	4	5	6	7
count_a=0	0	0	1	1	1	2	3	3
ans =0	0	0	0	1	2	2	2	5

</ > Code

total pairs

how many a's till now

```
ans = 0        count_a = 0
for( i: 0 → n-1) {
    if( s[i] == 'a')
        count_a ++
    else if( s[i] == 'g')
        ans += count_a
}
return ans
```

TC: O(n)

SC: O(1)

Subarrays

Continous part / continuous slice of array

4, 1, 2, 3, -1, 6, 9, 8, 12

0 1 2 3 4 5 6 7 8

Q → 4 -1 6 9 is sub-array? no

not continous

0 1 2 3 4 5 6 7

Example: arr[ ] → [2 4 1 6 -3 7 8 4]

6, 6

6, 7

a. [1, 6, 8]

b. [1, 4]

c. [6, 1, 4, 2]

d. [7, 8, 4,]

Representation of a subarray

start_idx end_idx

start_idx ≤ end_idx

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

s, e

0,0
0,1
0,2
0,3
0,4
0,5
0,6

ans = 7

1,1
1,2
1,3
1,4
1,5
1,6

ans = 6

Total number of subarrays

0, 1, 2, 3, ------ n-1

[4 2 10 3 12 -2 15] ,N = 7

0 1 2 3 4 5 6

Start at 0 ⇒ n

Start at 1 ⇒ n-1

Start at 2 ⇒ n-2

⋮

Start at n-1 ⇒ 1

total subarrays

= n + n-1 + n-2 + ---- + 1

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

< *Question* > : Given an array, si and ei. Print from si to ei.

si ≤ ei

arr → [4 2 10 3 12 -2 15] si = 2, ei = 5

0 1 2 3 4 5 6

10, 3, 12, -2

- void printSubarray(arr, si, ei) {

```
for( i: s → e ) {
|    print (arr[i])
}
```

}

print 1 subarray → T.C 0(N)

< *Question* > : Print all the possible sub-arrays of the given array.

[5, 7, 3, 2]

0 1 2 3

O/P - [5] 0, 0

[5, 7] 0, 1

[5, 7, 3] 0, 2

[5, 7, 3, 2] 0, 3

[7] 1, 1

[7, 3] 1, 2

[7, 3, 2] 1, 3

[3] 2, 2

[3, 2] 2, 3

[2] 3, 3

Idea Consider all the subarrays & print Subarray( )

start end

2 2

< / > Code

```
for ( i : 0 → n-1 ) {
    for ( j : i → n-1 ) {
        printSubarray (arr, i, j)
    }
}
```

TC: $O(N^3)$

i=0 j=2 0, 1, 2

i=1 2 1, 2

2 2 2

CRED

Min Max

< *Question* > : Given an array of N integers, return the length of smallest subarray which contains both maximum and minimum elements of the array.

$1 \le N \le 10^6$

arr[ ] → [2 2 6 4 5 1 5 2 6 4 1]

0 1 2 3 4 5 6 7 8 9 10

max = 6
min = 1

arr[ ] → [1 2 3 1 3 4 6 4 6 3]

0 1 2 3 4 5 6 7 8 9

max = 6
min = 1

arr[ ] → [8 8 8 8 8 8]

0 1 2 3 4 5

max = 8
min = 8
ans = 1

Idea

I will check all subarrays & get the min length where both max & min exist

Observation

1. There must be exactly one occurrence of min & max element.

   [min min - - - - max]

2. Min and max elements should be the end point of subarray.

   1 7 8 12 5

3. *case-1*: [min - - - - max]

   case-2: [max - - - - min]

arr[ ] → [2 2 6 4 5 1 5 2 6 4 1]

	0	1	2	3	4	5	6	7	8	9	10
latest_max=-1	-1	-1	2	2	2	2	2	2	8	8	8
latest_min=-1	-1	-1	-1	-1	-1	5	5	5	5	5	10

ans = 4 ~~3~~

2 3 4 5

abs(x) → if x ≥ 0 nothing

if x < 0 multiply with -1

< / > Code

1) min_val & max_val

latest_max = -1 latest_min = -1

ans = INT_MAX (max value that 32 bit int can store

```
for( i: 0 ⟶ n-1 ) {
    if( arr[i] == minval ) {
        latest_min = i
    }
    if( arr[i] == max_val ) {
        latest_max = i
    }
    if ( latest_max != -1 && latest_min != -1){
        ans = min ( ans, abs( latest_max - latest_min) +1)
    }
}
return ans
```

TC: O(n)

SC: O(1)

$r - l + 1$

$[l, r]$

$r - l + 1$