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'.

$$s = \text{"}\begin{array}{cccccc}0&1&2&3&4&5\\a&b&e&g&a&g\end{array}\text{"}$$

Ans = 3

i	<	j
0		3
0		5
4		5

$$s = \text{"}\begin{array}{ccccccc}0&1&2&3&4&5&6\\a&c&g&d&g&a&g\end{array}\text{"}$$

Ans = 4

i	<	j
0		2
0		4
0		6
5		6

$$s = \text{"}\begin{array}{cccccccc}0&1&2&3&4&5&6&7\\b&c&a&g&g&a&a&g\end{array}\text{"}$$

Ans = 5

i	<	j
2		3
2		4
2		7
5		7
6		7

BF Idea

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

TC = $\underline{O(N^2)}$ SC = $\underline{O(1)}$

```
cnt = 0
for i → 0 to (N-2) {
    if (s[i] == 'a') {
        for j → i+1 to (N-1) {
            if (s[j] == 'g')
                cnt++
        }
    }
}
} return cnt
```

count #'g' from (i+1) to (N-1).

ans += cnt_g[i+1]

TC = $\underline{O(N^2)} \rightarrow \underline{O(N)}$ SC = $\underline{O(1)} \rightarrow \underline{O(N)}$

"0 1 2✓ 3 4 5✓ 6✓ 7"

s = "b c a g g a a g"

L ← R

cnt_g → [3 3 3 3 2 1 1 1]

ans = 0 + 3 + 1 + 1 = $\underline{5}$

```
∀i, if (s[i]=='g')
        cnt_g[i] = cnt_g[i+1]+1
    else  // s[i] ≠ 'g'
        cnt_g[i] = cnt_g[i+1]
```

Idea Carry Forward → calculate & use

	0	1	2	3	4	5	6	7
str →	b	c	a	g	g	a	a	g
'g' → cnt	3	3	3	3	2	1	1	1

L ← R

'a' → ans = 0 + 1 + 1 + 3 = 5

```
cnt = 0
ans = 0
for i → (N-1) to 0 {
    if (s[i] == 'g') cnt++       // calculate
    if (s[i] == 'a') ans += cnt  // use
}
```

TC = O(N)   SC = O(1)

i < j   s[i] = 'a'   s[j] = 'g'

count # 'a' from left to right 0 —— i

c a a g x g

cnt = + 2

ans = 2 + 2 = 4

```
cnt = 0
ans = 0
for i → 0 to N-1 {
    if (s[i] == 'a') cnt++       // calculate
    if (s[i] == 'g') ans += cnt  // use
}
```

TC = O(N)   SC = O(1)

Subarrays

→ continuous part of array

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & \\ A = [& 1 & 2 & 3 & 4 & 5 &] \end{array}$$

single element ✓

complete array ✓

Example: arr[ ] → [2 4 1 6 -3 7 8 4]

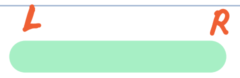
a. [1, 6, 8]

b. [1, 4]

c. [6, 1, 4, 2]

d. [7, 8, 4,] ✓

Representation of a subarray

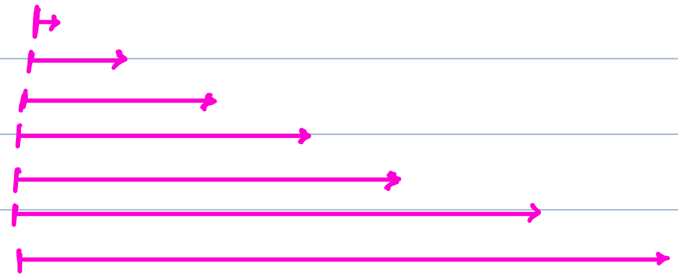
1) 

start & end

2) L (length) start & length

0 1 2 3 4 5 6

A = [4 2 10 3 12 -2 15]

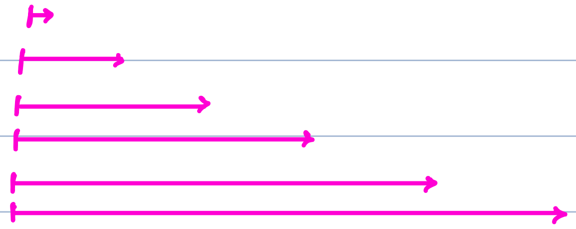


subarrays = 7

subarrays starting from index 0 = N

0 1 2 3 4 5 6

A = [4 2 10 3 12 -2 15]



subarrays = 6

Total number of subarrays

subarrays starting from ↓ = ↓

0	N
1	N-1
2	N-2
⋮	
(N-1)	1
	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

o/p → 10 3 12 -2

```
• void printSubarray( arr, si, ei ) {

      for i → si to ei {
          print ( arr [i] )
      }

  }
```

print 1 subarray → T.C 0(N)

SC = O(1)

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

[5, 7, 3, 2]

0 1 2 3

O/P - [5]

[5, 7]

[5, 7, 3]

[5, 7, 3, 2]

[7]

[7, 3]

[7, 3, 2]

[3]

[3, 2]

[2]

$\frac{N * (N+1)}{2} * N \rightarrow \underline{O(N^3)}$

$\boxed{8:30 \text{ AM}}$

Idea Consider all the subarrays & print Subarray( )

```
for st → 0 to (N-1) {
    for end → st to (N-1) {
        for i → st to end {
            print (A[i])
        }
        print ('\n')
    }
}
```

$TC = \underline{O(N^3)}$ $SC = \underline{O(1)}$

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

Ans = 3

arr[ ] → [1 2 3 1 3 4 6 4 6 3]

0 1 2 3 4 5 6 7 8 9

Ans = 4

arr[ ] → [8 8 8 8 8 8]

0 1 2 3 4 5

min = 8 max = 8 Ans = 1

Bruteforce → Find min & max of the array. → O(N)

∀ subarrays, check if it contains min & max and take smallest length subarray as answer. O(N)

$\frac{N*(N+1)}{2}$

$TC = O(N + N^3) = O(N^3)$ $SC = O(1)$

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.

x min max

3. case-1: [min - - - - max] → ∀ min check closest max on right

✓ ∀ max check closest min on left

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

✓ ∀ min check closest max on left

∀ max check closest min on right

	0	1	2	3	4	5	6	7	8	9	10	
arr[ ] → [	2	2	6	4	5	1	5	2	6	4	1	]
	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	i	

min = 1

max = 6

latest_min_index = ~~-1~~ ~~5~~ 10

latest_max_index = ~~-1~~ ~~2~~ 8

length = 10 - 8 + 1 = 3

ans = ~~4~~ 3

</> Code

```
minA = A[0]          maxA = A[0]
for i → 1 to (N-1) {
    minA = min (minA, A[i])
    maxA = max (maxA, A[i])
}
l_min = -1          l_max = -1
ans = N
for i → 0 to (N-1) {
    if (A[i] == minA) {
        l_min = i                      //calculating
        if (l_max != -1) {
            ans = min (ans, i - l_max + 1)
        }                                          // use
    }               [l_max          i]
    if (A[i] == maxA) {
                                          min ———— max
        l_max = i                        [l_min ———— i]
        if (l_min != -1) {
            ans = min (ans, i - l_min + 1)
        }
    }
}  return ans              TC = O(N)      SC = O(1)
```