Carry Forward & Subarrays

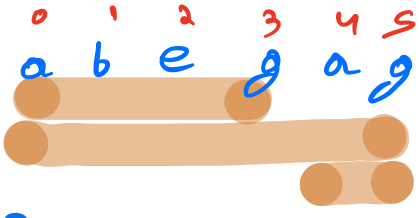
⇒ Intro about contest

- ⇒ 9:00 → 10:30 (1.5 hour) 10:30 → 11:30 (Doubt Session)
- ⇒ 3 questions [2 questions to pass]
- ⇒ contest will be Monday [re-attempt]

Question 1 :- Given a string 's' of lowercase characters, return the count of pairs (i,j) such that

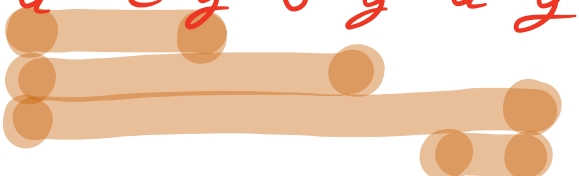
i<j & s[i] is 'a' & s[j] is 'g'.

Ex :- s = "abegag"



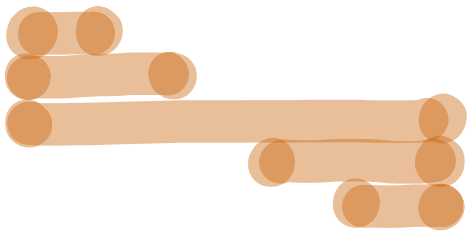
Ans = 3

Quiz 1 s = "acgdgag"



Ans = 4

Quiz 2 $S = "b\ c\ a\ g\ g\ a\ a\ g"$



Ans = 5

Brute force

↳ For each 'a' we can check how many 'g' are present on right. And For this we can use two nested loop.

PSEUDO CODE

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

TC → O(N * N)

SC → O(1)

↳ OPTIMIZATION

Observation

① For each 'g'. Count the nos. of 'a' on the Left.

② We can maintain a counta variable for storing the no. of 'a' on the left Side while iterating from Left to Right.

DRY RUN

					↓			↓	↓
	a	c	b	a	g	k	a	g	g
counta =	1	1	1	2	2	2	3	3	3
ans =	0	0	0	0	2	2	2	5	8

PSEUDO CODE

```
int ans = 0;
int counta = 0;
for (i = 0; i < n; i++){
    if (s[i] == 'a'){
        counta++;
    }
    if (s[i] == 'g'){
        ans += counta;
    }
}
```

return ans;

TC → O(N)
SC → O(1)

★ This technique is known as **carry forward**

SUBARRAYS

A Subarray is a **contiguous part** of an array. It is formed by **deleting a range of elements from the array**. A Subarray can have one or more elements & must be a **contiguous part** of the original array

Ex:- arr[] = [4, 1, 2, 3, -1, 6, 9, 8, 12]

[2 3 -1 6] ✓

[9] ✓

[4, 1, 2, 3, -1, 6, 9, 8, 12] ✓

[4, 12] ✗

[1, 2, 6] ✗

[3 2 1 4] ✗

Quiz3 :- A[] = {2, 4, 1, 6, -3, 7, 8, 4}

Ans => {7 8 4}

Representation of an Subarray.

↳ Two Way

arr[] = [4(0), 1(1), 2(2), 3(3), -1(4), 6(5), 9(6), 8(7), 12(8)]

Ex:- [2 3 -1 6] —First Way→ (2 to 5)

(2 to 5): 2 = start index, 5 = end index

[2 3 -1 6] —Second Way→ (2 & 4)

(2 & 4): 2 = start index, 4 = length of Subarray

(II) [9] —First Way→ (6 to 6)

[9] —Second Way→ [6 & 1]

Quiz 4 How many Subarrays of the following array starts from index 0!

[4(0), 2(1), 10(2), 3(3), 12(4), -2(5), 15(6)]

Ans

	index
[4]	(0,0)
[4 2]	(0,1)
[4 2 10]	(0,2)
[4 2 10 3]	(0,3)
[4 2 10 3 12]	(0,4)
[4 2 10 3 12 -2]	(0,5)
[4 2 10 3 12 -2 15]	(0,6)

Quiz 5 :- How many subarrays of the following array starts from index 1 ?

$[4^0, 2^1, 10^2, 3^3, 12^4, -2^5, 15^6]$

Ans

	index
[2]	(1,1)
[2 10]	(1,2)
[2 10 3]	(1,3)
[2 10 3 12]	(1,4)
[2 10 3 12 -2]	(1,5)
[2 10 3 12 -2 15]	(1,6)

Formula to count total subarrays

arr[ ]

index → 0, 1, 2, 3, ... n-2, n-1

Subarrays starting with 0	Subarray starting with 1	Subarray starting with 2
(0,0)	(1,1)	(2,2)
(0,1)	(1,2)	(2,3)
(0,2)	(1,3)	(2,4)
⋮	⋮	⋮
(0,n-1)	(1,n-1)	(2,n-1)
Total n	n-1	n-2

Subarray starting with (n-1)

(n-1, n-1)

1

Total Subarrays

$$\Rightarrow N + (N-1) + (N-2) + \ldots\ldots + 1$$

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

Ex:- [1 2 3]

Total Subs :-

[1]
[1 2]
[1 2 3]
[2]
[2 3]
[3]

(6)

By formula

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

$$\frac{3*4}{2} = 6$$

Question2:- Given an array of integers & 2 indices, a start & End index. we need to print the subarrays of the array that starts from start index & ends at the end index (both inclusive)

Ex:- [5 4 3 1 2]
0 1 2 3 4

S = 1
E = 3

Ans:- [4 3 1]

Brute Force

↳ We can print from S to E

PSEUDO CODE

```
void printSubarray ( A[], S, E ){
    for (i = S; i <= E; i++){
        Print (A[i]);
    }
}
```

TC → O(N)
SC → O(1)

Question 3 :- Print all the subarrays of the given array.

Ex :- [10 , 20 , 30]
0 1 2

Subarrays :-

[10]
[10 20]
[10 20 30]
[20]
[20 30]
[30]

OBSERVATION

① First of all we need find all possible S & E. For getting all the possible S & E, we can use nested loops.

PSEUDO CODE

```
for (s=0 ; s<n; s++){
    for (e=s; e<n; e++){
        print (s, e);
    }
}
```

DRY RUN

s	e
0	0
0	1
0	2
1	1
1	2
2	2

(2) We just Need to iterate from s to e. And print it.

PSEUDO CODE

```
for (s=0; s<n; s++){
    Pick (s,e)
    for (e=s; e<n; e++){
        for (i=s; i<=e; i++){
            Print (A[i]);
        }
    }
}
```

(Printing from s to e)

$TC \rightarrow O(N^3)$

$SC \rightarrow O(1)$

Question 4 Given an array of N integers, return the length of smallest subarray which contains both maximum & minimum elements of the array.

Quiz 6 which is the length of the smallest subarray which has both, the Max & Min of the array?

Ex:- $\overset{0}{2}, \overset{1}{2}, \overset{2}{6}, \overset{3}{4}, \overset{4}{5}, \overset{5}{1}, \overset{6}{5}, \overset{7}{2}, \overset{8}{6}, \overset{9}{4}, \overset{10}{1}$

Max = 6 Min = 1

$[\overset{8}{6}\ \overset{9}{4}\ \overset{10}{1}]$ A

(3)

Brute force

↳ check all the possible subarrays & the one with minimum length containing Max & Min will be the Ans.

★C HW)

TC → O(N³)

SC → O(1)

↳ OPTIMIZATION

Observation

① We need only 1 Max & 1 Min in Ans.

② We will be having only Max & Min on the corners

- → Max on Left & Min on Right
- → Min on Left & Max on Right

③ Basically we are looking for subarray which starts with Max & end on closest possible Min or vice-versa

④ We can keep a track of last closest Max & Min to check smallest possible Sub.

DRY RUN

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

Let,
last_Min_idx = -1
last_Max_idx = -1
ans = INT_MAX
minValue = 1
maxValue = 6

i	A[i]	last_min_idx	last_max_idx	ans
0	2	-1	-1	∞
1	2	-1	-1	∞
2	6	-1	2	∞
3	4	-1	2	∞
4	5	-1	2	∞
5	1	5	2	5-2+1=4
6	5	5	2	4
7	2	5	2	4
8	6	5	8	8-5+1=4
9	4	5	8	4
10	1	10	8	10-8+1=3

PSEUDO CODE

```
int max  // TODO
int min  // TODO
last_min_idx = -1, last_max_idx = -1
ans = ∞

for (i = 0; i < n; i++) {
    if (arr[i] == max) {
        if (last_min_idx != -1) {
            ans = min(ans, i - last_min_idx + 1)
        }
        last_max_idx = i;
    }
    else if (arr[i] == min) {
        if (last_max_idx != -1) {
            ans = min(ans, i - last_max_idx + 1);
        }
        last_min_idx = i;
    }
}

return ans;
```

↳ EDGE CASE :- 888 I all elements same

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

TC → O(N) SC → O(1)

Next Class Content

- SLIDING WINDOW
- CONTRIBUTION TECHNIQU
-