Contest

→ 1.5 hr contest with 3 questions.

→ Online Assessment works for companies.

→ Test Yourself

→ If you are doing regular assignments then a single revision.

Q: Given a string s of lowercase characters, return the count of pairs (i, j) such that

(1) $i < j$

(2) $s[i] == \text{'a'}$ and $s[j] == \text{'g'}$

```
         0 1 2 3 4 5
Ex:  s = " a b e g a g "
```

Ans: 3

(0 3)
(0 5)
(4 5)

```
                  0 1 2 3 4 5 6
Quiz 1: Count in " a c g d g a g "
```

Ans: 4

0 2
0 4
0 6
5 6

Quiz 2: Count in "bcaggaag"
(indices 0 1 2 3 4 5 6 7)

Ans: 5

2	3
2	4
2	7
5	7
6	7

Brute Force:

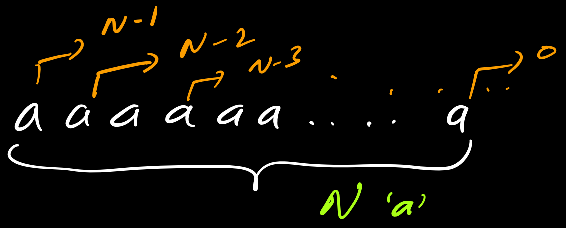
For every 'a', we need to find the count of g on the right side of a.

=) Nested Loops

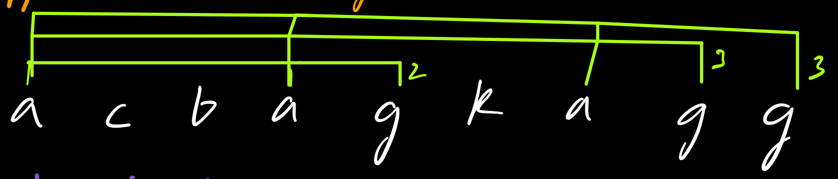
```
int countAg(String s) {
    int ans = 0
    for(int i = 0; i < N; i++) {
        if(s[i] == 'a') {
            for(int j = i+1; j < N; j++) {
                if(s[j] == 'g')
                    ans++
            }
        }
    }
    return ans;
}
```

TC: $O(N^2)$

SC: $O(1)$



Optimised Approach: Carry Forward



		a	c	b	a	g	k	a	g	g
count A	0	1	1	1	2	2	2	3	3	3
						↓+			↓+	↓+
ans	0	0	0	0	0	2	2	2	5	8

ans = 8

```
int countAg (String s) {
    int ans = 0
    int countA = 0
    for(int i = 0; i < N; i++) {
        if(s[i] = = 'a') {
            countA ++;
        }
        else if ( s[i] = = 'g') {
            ans + = countA;
        }
    }
    return ans
}
```

TC: O(N)

SC: O(1)

Intro to Subarrays

- → A subarray is a contiguous part of an array
- → It is formed by selcting a range of elements from the array.
- → It can have one or more elements but it must be contiguous part of orignal array.

4	1	2	3	-1	6	9	8	12

2 3 −1 6 is a subarray of size 4

9 is a subarray of size 1

4 1 2 3 −1 6 9 8 12 in a subarray with all elements

4 12 is not a subarray as not contiguas

1 2 6 is not a subarray as not contiguas

3 2 1 4 is not a subarray as order is different

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

(A) 1 6 8 X

(B) 1 4 X

(C) 6 1 4 2 X

(D) 7 8 4 YES

Representation of Subarray

0	1	2	3	4	5	6	7	8
4	1	2	3	-1	6	9	8	12

The subarray 2, 3, -1, 6 can be represented by

1) Start & End Index 2 to 5 (start and end index of subarray)

2) Start and Size start = 2
size = 4

Q How many subarrays starting from index 0?

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

start end

(0, 0) (0, 4)

(0, 1) (0, 5) ans = 7

(0, 2) (0, 6)

(0, 3)

Q How many subarrays starting from index 1 ?

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

(1, 1)   (1, 3)   (1, 5)   ans = 6

(1, 2)   (1, 4)   (1, 6)

Formula to count total No. of subarrays

No. of subarrays starting from index 0 = N

" " " " " " 1 = N-1

" " " " " " 2 = N-2

⋮

No. of subarray starting from index N-2 = 2

" " " " " " N-1 = 1

Sum of N natural numbers = $\frac{N * (N+1)}{2}$

Q Print the subarray of the array that starts from the given start index and end at the given end index.

0	1	2	3	4	5
1	2	-1	2	5	6

start = 1
end = 3

output
2 -1 2

```
void printSubarray( int arr[], int start, int end){
        for( i = start; i <= end; i++) {
              print( arr[i] );
        }
        print('\n');
}
```

TC: O(N)

SC: O(1)

Q Print all possible subarrays of the array.

[1 2 3]

output

[1]

[1 2]

[1 2 3]

[2]

[2 3]

[3]

$$\frac{3 * (3+1)}{2} = \frac{3 \times \cancel{4}^{2}}{\cancel{2}}$$

$$= 6$$

```
Void printAllSubarrays( int arr[], int n){
    for( int i = 0 ; i<N ; i++){        i is start
        for( int j = i ; j<N ; j++){        j is end
            # Print the subarray -> (i, j)
            for( int K = i; K<=j ; k++){
                print(arr[K]);
            }
            print('/n');
        }
    }
}
```

printSubarray(arr, i, j)

TC: $O(N^3)$ SC: $O(1)$

1 2 3 4 5

i j

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

2

2 3

2 3 4

2 3 4 5

3

3 4

3 4 5

4

4 5

5

$\frac{5*6}{2} = 15$

Q Given an array of N integers, return the len of smallest subarray which contains both maximum and minimum element of the array.

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

max = 6

min = 1

Brute Force : 1) Find Max and Min TC: O(N)

Naive

Explore all

2) Explore all subarray and for subarray which has both max and min. TC: $O(N^3)$

Consider them for ans.

TC: $O(N^3)$ SC: O(1)

Observations :

1) Max and Min can come in any order.

2) MAX . . . . MIN

MIN . - - — . - . MAX

=) Iterate if found :

MAX => then go ahead and find next min.

MIN => then go ahead and find next max.

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

max = 6

min = 1

ans = ~~11~~ ~~4~~ 3

TC: $O(N + N^2)$ SC: $O(1)$

$\approx O(N^2)$

Carry Forward :

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

=> Calculat MAX MIN beforehand

i	A[i]	last_min_index	last_max_index	ans
–	–	–1	–1	11 / INT_MAX
0	2	–1	–1	11
1	2	–1	–1	11
2	6	–1	2	11
3	4	–1	2	11
4	5	–1	2	11
5	1	5	2	$(5-2+1) = 4$ 4

6	5	5	2		4
7	2	5	2		4
8	6	5	8	(8-5)+1 =4	4
9	4	5	8		
10	1	10		(10-8)+1 = 3	3

```
int  minSubarray( int  A[], int  n){

    int min = minOfArray( A);
    int max = maxofArray (A);

    int last_min_index = -1
    int last_max_index = -1
    int  ans = N

    for(i=0; i<N, i++){

        if(A[i] == minValue){
            last_min_index = i
            if( last_max_index != -1){
                ans = min(ans, i- last_max_index +1);
            }

        }
        else      if(A[i] == maxValue){
            last_max_index = i
            if( last_min_index != -1) {
                ans = min(ans, i - last_min_index +1);
            }

        }
    }
    return ans;
}
```

TC: O(N)

SC: O(1)

Next Class:

-> Subarrays
-> Printing all subarrays
-> Contribution Technique

Doubts

0 based Index

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

prefix Array

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

sum=> 2 5 => pf(5) - pf(1) = 10 - 3 = 7

sum => 0 3 => pf[3] = 7

To avoid if else on the l == 0

1 based Index

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

prefix Array

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

sum=> 2 5 => 3 6 => pf[6] - pf[2] = 10 - 3 = 7

sum => 0 3 => 1 4 => pf[4] - pf[0] = 7 - 0 = 7

1 2 3 4

qu 1 based

1 2 => 3