Carry forward Subarray ⇒ Define
⇒ Iterate
⇒ Problem

Given a string S of lowercase alphabets

Return the count of pairs (i, j) s.t.

1) $i < j$
2) $S[i] = 'a' \neq S[j] = 'g'$

Eg: S = "a b e g a g"
0 1 2 3 4 5

i	j
0	3
0	5
4	5

} Ans = 3

S = "a c g d g a g"
0 1 2 3 4 5 6

(0, 2), (0, 4), (0, 6), (5, 6)

Ans = 4

S = 0 1 2 3 4 5 6 7
b c a g g g a a g

(2,3) (2,4) (2,7) (5,7) (6,7)

Ans = 5

Sol^n > Brute force ⇒ Check for. All possib.

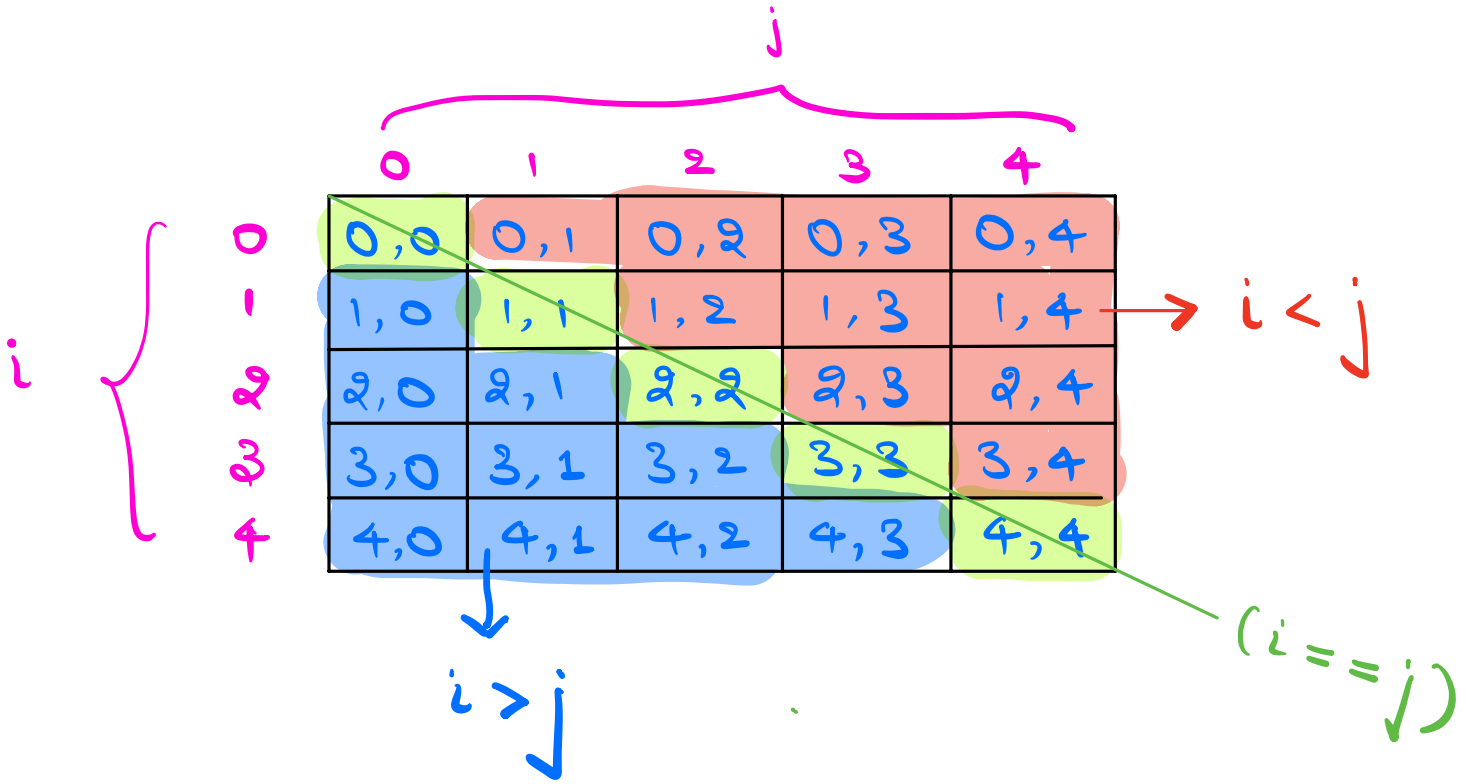
⇒ Check for all pairs the 2 given condi. ∉ count++;

Code

```
count = 0;
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if((i < j) && S[i] == 'a' && S[j] == 'g') {
            count++;
        }
    }
}
```

T.C. = $O(N^2)$
S.C. = $O(1)$

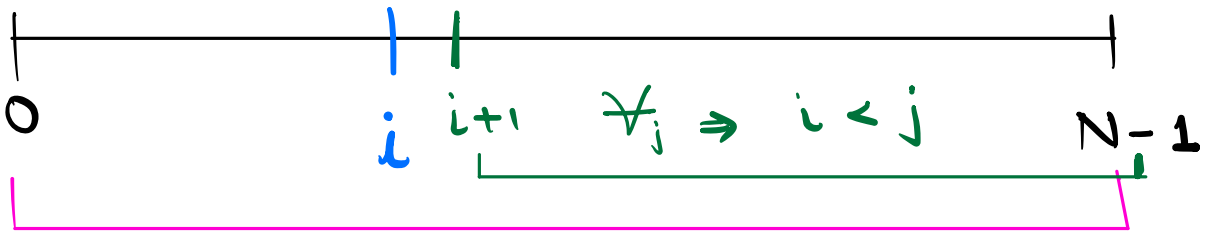
A: [2, 3, 4, 5, 1]



Intro to Array ⇒ Good pair.

Pair (i, j) s.t. (A[i] + A[j] == K)

(i, j) = (j, i)



⇓

∀ possible j

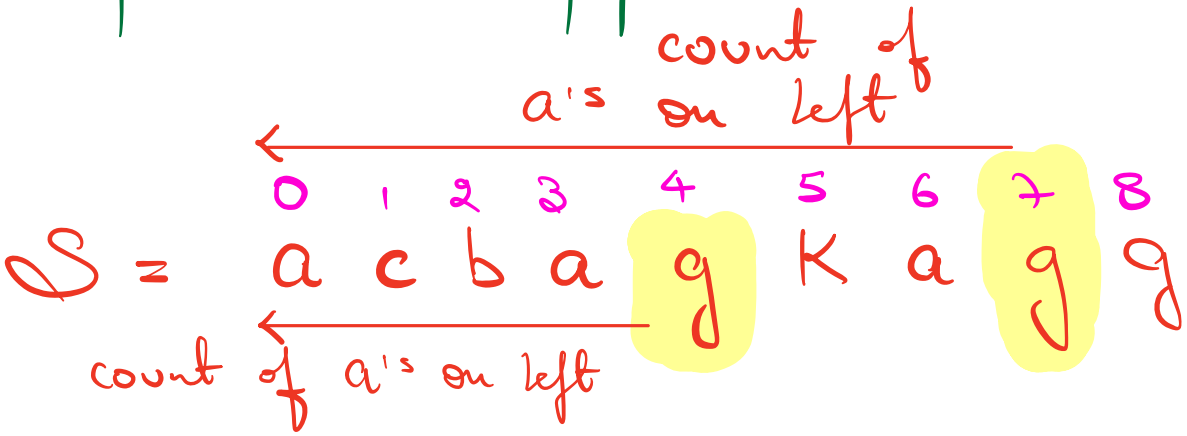
∀ → for all

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

T.C. = $O(N^2)$

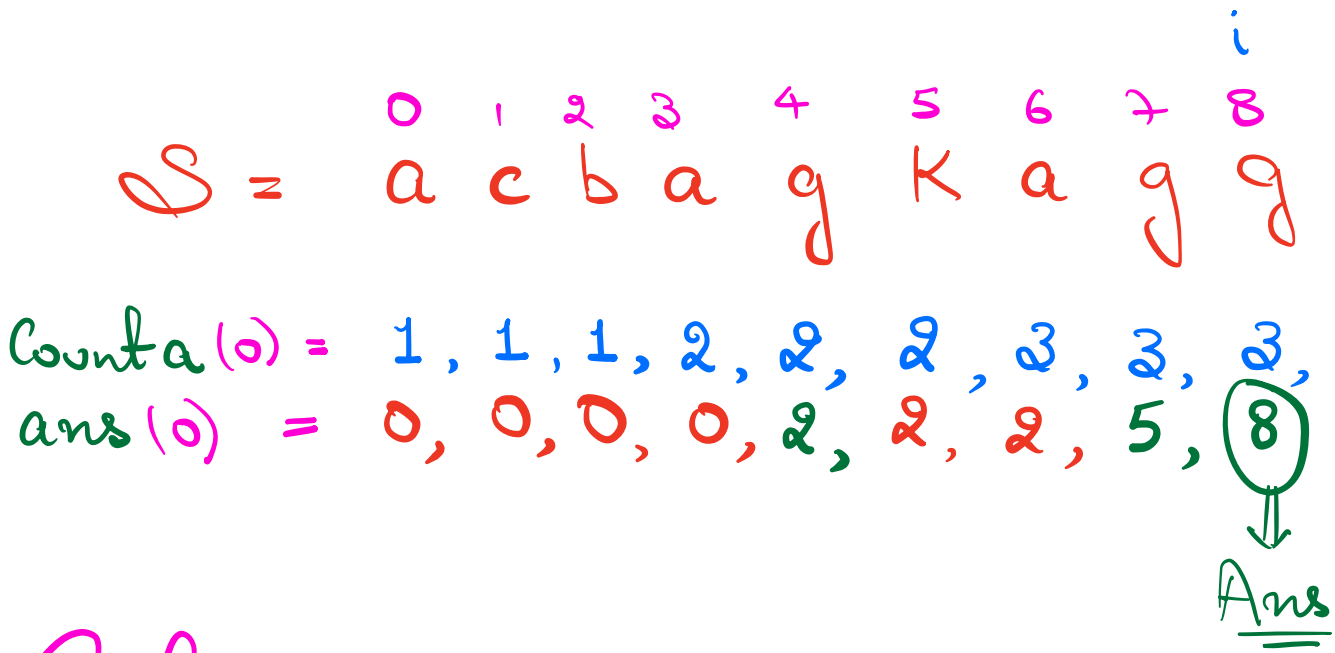
S.C. = $O(1)$

2) Optimised Approach



Observations

1) for every 'g' we need to know the count of 'a' in the left



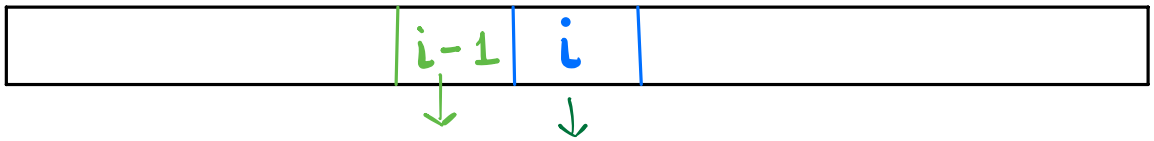
Code

```
ans = 0;
counta = 0;

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

        if ( S[i] == 'a' ) {
                counta++;
        }
        else if ( S[i] == 'g" ) {
                ans = ans + counta;
        }
}
return ans;
```

T.C. = O(N)
S.C. = O(1)

This technique is known as Carry forward.

Carry forward



count of a in S[0, i]

↳ 1) Iterate from 0 to i

↳ 2) Use count a in S[0, i-1]

Carry forward. ⇐

If (s[i] == a) else

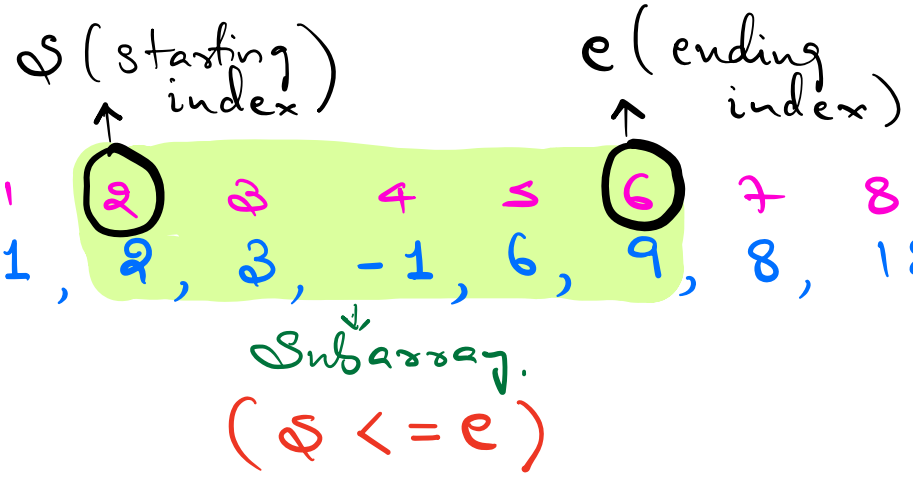
count a[i] = count a[i-1] + 1 | count a[i] = count a[i-1]

Subarray

⇒ Continuous part of an array.

⇒ A subarray can be defined by selecting a valid range of index in the array.

⇒ A subarray must have one of more elements present in it.



Eg 1) [2, 3, -1, 6, 9] ⇒ Subarray of length 5

2) [9] ⇒ Subarray with 1 element (s == e)

3) [12, 4, 1] ⇒ Subarrays are not circular

4) [1, 2, 6] ⇒ Not subarray as not conti.

5) [3, 2, 1, 4] ⇒ No, as order of elements is different

Count of elements in a subarray from index s to index e.

$$\text{Count} = e - s + 1 = \text{length of subarray}$$

Length of subarray from s to e $= e - s + 1$

A = {2, 4, 1, 6, -3, 7, 8, 4}

1) [1, 6, 8] ✗

[1, 4] ✗

[6, 1, 4, 2] ✗

[7, 8, 4] ✓

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

(0 1 2 3 4 5 6)

s ≤ e

s: 0 0 0 0 0 0 0 1 1

e: 0 1 2 3 4 5 6 1 2

= [4]

= [4, 2]

= [4, 2, 10]

= [4, 2, 10, 3]

= [4, 2, 10, 3, 12]

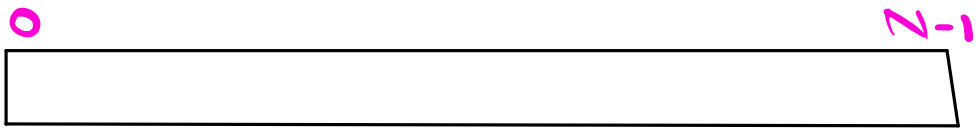
= [4, 2, 10, 3, 12, -2]

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

1	3
1	4
1	5
1	6

Ans = 6

Count of subarrays for a given array of size N.



S	C	Count
0	[0, N-1]	N
1	[1, N-1]	N-1
2	[2, N-1]	N-2
3	[3, N-1]	N-3
⋮		⋮
N-1	[N-1, N-1]	1

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

Total no. of subarrays for an array of size N $= \frac{(N)(N+1)}{2}$

Q. Given an integer array of size N. & 2 integers s & e.

Print the subarray from index s to e.

Sol^{n} Iterate from s to e & print

Code

```
void printSubarray (A[], s, e) {
    for (i = s; i <= e; i++) {
        print (A[i]);
    }
}
```

T.C. = O(N)

Q. Given an integer array of size N. Print all subarrays.

A = [1, 2, 3] (0, 1, 2)

s = 0
[1] e = 0
[1, 2] e = 1
[1, 2, 3] e = 2

s = 1
[2] e = 1
[2, 3] e = 2

s = 2
[3] e = 2

Code

```
for (s = 0;  s < N;  s++) {
    for (e = s;  e < N;  e++) {          } N^2
        // Subarray from s to e.
        N  ⇐ printSubarray(A, s, e);
    }
}
```

T.C. = $O(N^3)$

s	e	
0	0	⇒ A[0] = 1
0	1	⇒ A[1] = 2 ✗
0	2	⇒ A[2] = 3 ✗
1	1	
1	2	
2	2	

Given an integer array of size N. Return the length of the smallest subarray which contains both the

mex & min. 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

⇓

Ans = 2

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

mex, min

Q Does subarray [2, 6] also contains both mex & min ??

Obs: Smallest subarray will always have mex & min as boundary elements.

Soln 1) Brute force (All possi.)

Calculate mex & min. // O(N)

⇒ ∀ subarrays ⇒ > Check if it contains both mex & min // O(N)

O(N²)

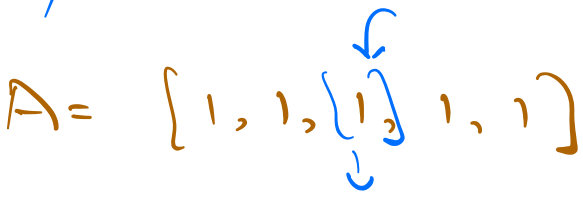
2) If yes ⇒ compare length & take min

T.C. = $O(N^3)$
S.C. = $O(1)$

H.W. ⇒ Implement.

2) Optimisation

Obs: Smallest subarray starts & ends on max / min.

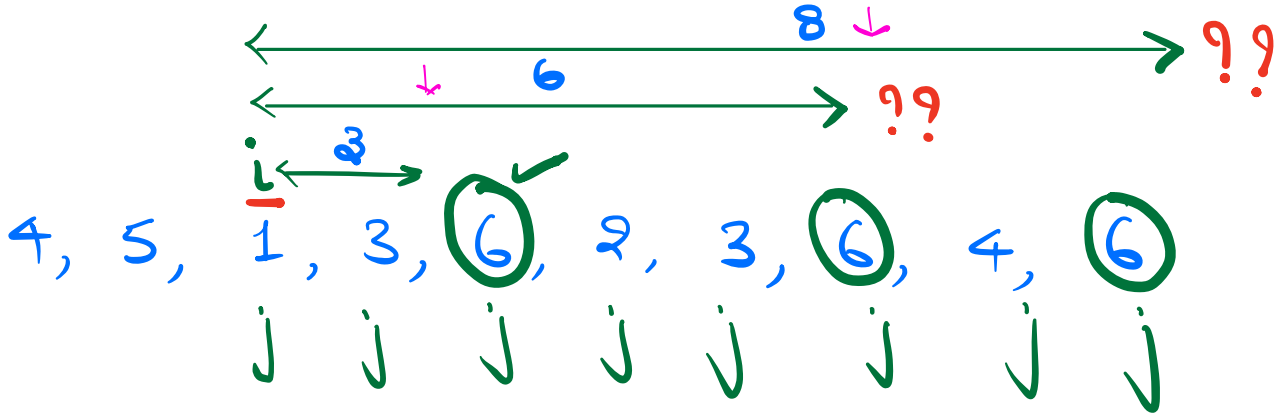


Code

```
ans = INT_MAX ;

for (s = 0;  s < N;  s++) {  //

    if ( A[s] == min) {
        for (e = s;  e < N;  e++) {  //
            if (A[e] == max) {
                l = e - s + 1;
                ans = min (ans, l);
            }
        }
    }
    else if ( A[s] == max) {
```

```
for (e = s; e < N; e++) {
    if (A[e] == min) {
        l = e - s + 1;
        ans = min(ans, l);
    }
}
}
}
```

T.C. = $O(N^2)$



Obs : for min ⇒ nearest mex in right

for mex ⇒ nearest min in right

H.W.

⇓

Carry forward

T.C. = O(N)

S.C. = O(1)

Next Class ⇒ Wednesday

Sat
Sun
Mon
Tuesday

1) Complete Assig = PSP=100
2) Complete all additional problems

⇓

Optional PS session

⇓

Doubts

min

Del → Bang

ans = INT_MAX

30
50
70
100
60
25
40

→ 30

min