Carry Forward & Subarray

Content

- Count of pairs ag
- Subarray
- Print all subarrays
- Max and Min

Count of pairs ag

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

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

String s =

0	1	2	3	4	5
a	b	e	g	a	g

a	g
0	3
0	5
4	5

ans = 3

s =

0	1	2	3	4	5	6
a	c	g	d	g	a	g

a	g
0	2
0	4
0	6
5	6

s =

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

a	g
2	3
2	4
2	7
5	7
6	7

Brute force

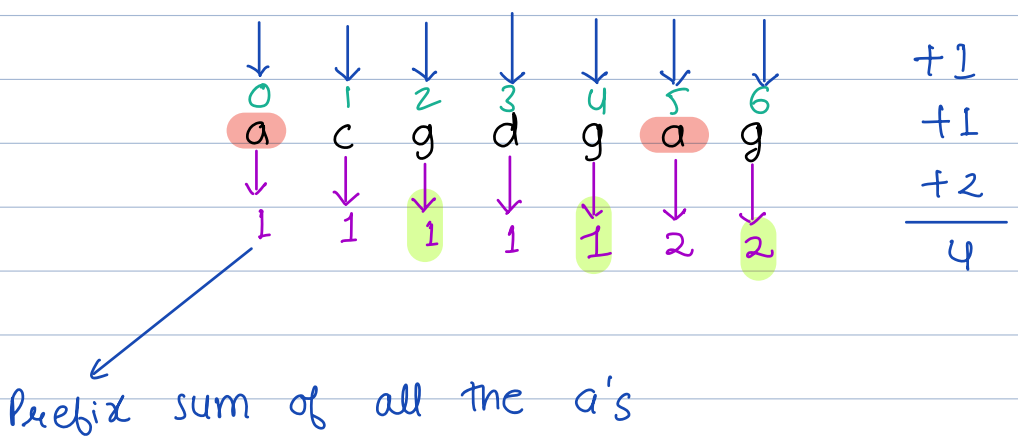
```
int count_ag ( string s ) {
    count = 0

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

TC : $O(N^2)$

SC : $O(1)$

Optimise the above solution ?



Optimised

// Create prefix arr to store the count of A

PA = [] ⟶ calculate

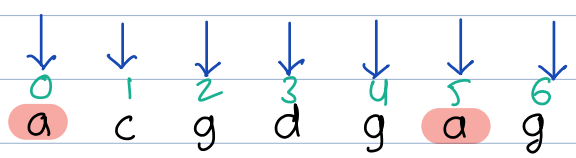
```
count = 0
for (i ⟶ 0 to N-1) {
    if (A[i] == 'g')
        count += PA[i]
}
print(count)
```

use

TC : O(N)

SC : O(N)

Can you optimise this ?



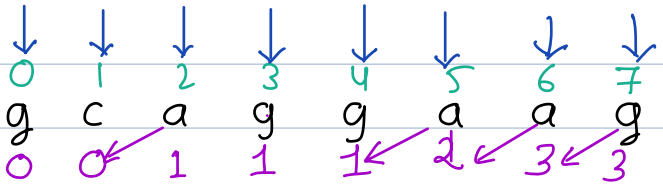
any +1 +1 +2 = 4

```
cnta = ∅ // 2
count = 0        cnta = 0
for (i ⟶ 0 to N-1) {
    if (A[i] == 'a') {cnta += 1}
    if (A[i] == 'g') {count += cnta}
}
print(count)
```

TC : O(N)

SC : O(1)

calculate & use carry forward.



cnta = ~~0~~ ~~1~~ ~~2~~ 3

count = 0 + 0 + 1 + 1 + 3 = 5

Carry Forward technique

⟶ Reusing the previously calculated value

Subarray

a continuous part of an array.

A = 4 1 2 3 −1 6 9 8 12

subarray of A

→ 4 1 2 3 ✓

3 −1 ✓

12 ✓

8 9 ✗

A = [2 4 1 6 −3 7 8 4]

~~1 6 8~~

~~1 4~~

~~6 1 4 2~~

7 8 4 ✓

Representation of subarray

Ways to represent a subarray.

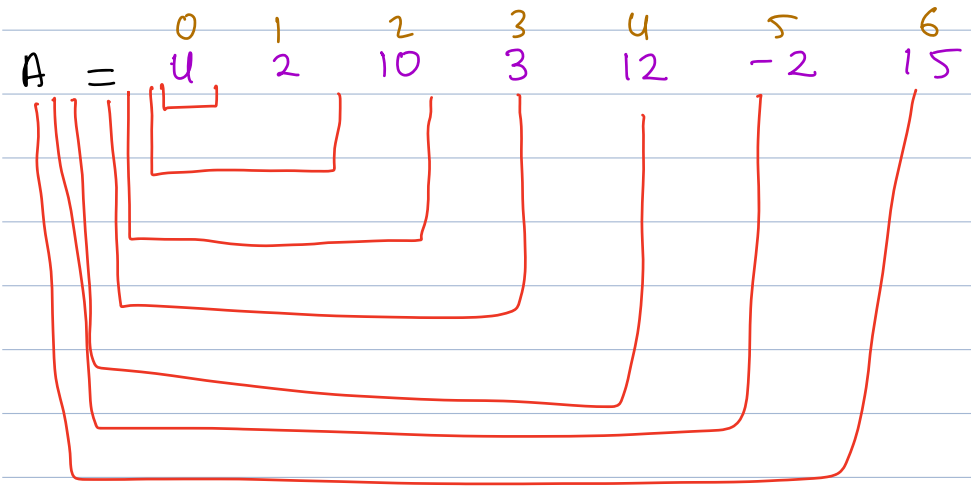
1> start = 2 end = 4

2> start = 1 length = 2

A = [1 2 3 4 5]
0 1 2 3 4

A = [1 2 3 4 5]
0 1 2 3 4

[2, 3].



0
4

0 1
4 2

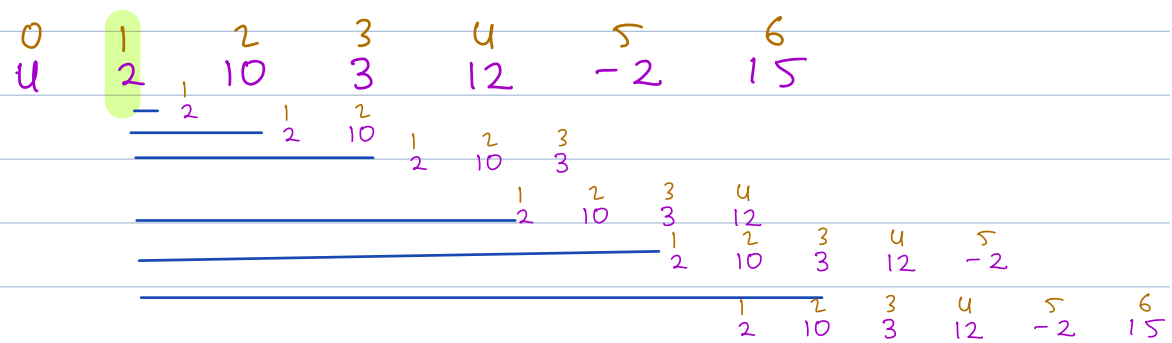
0 1 2
4 2 10

0 1 2 3
4 2 10 3

0 1 2 3 4
4 2 10 3 12

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

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



No. of subarrays of any array.

$A = [a_0 \quad a_1 \quad a_2 \quad \ldots\ldots \quad a_{n-1}]$ (indices 0, 1, 2, ..., n-1)

subarrays will start at index 0 → n

—————— 11 —————— 1 → n-1

⋮

n-1 → 1

Sum of all natural no. till n $= \frac{n*(n+1)}{2}$

No. of subarrays of an array of size n $= \frac{n*(n+1)}{2}$

Count of all subarrays.

Given an A[] and two indices start & end index we need to print the subarray of the array from start to end index.

A = 1 2 3 4 5 (indices 0 1 2 3 4) s = 1 e = 3

```
void printSub ( A[] start , end) {
    for ( i = start ; i <= end ; i++) {
            print ( A[i] )
    }
}
```

TC : O(N)

SC : O(1)

Given an int[] A , we need to print all posible subarrays of the array.

Input 1 : A = [1 2 3] $\frac{3*(3+1)}{2} = 6$

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

```
void printSubarrays ( A[] ) {
// To define a subarray I need start & end
for ( i = 0 ; i < n ; i++ ) {                    O(N)
    for ( j = i ; j < n ; j++ ) {                O(N)
        printSub ( A[], i, j )                   O(N)
    }
}
```

TC : $O(N^3)$

```
for ( i = 0 ; i < n ; i++ ) {                    O(N)
    for ( j = i ; j < n ; j++ ) {                O(N)
        for ( k = i ; k <= j ; k++ ) {
            print ( A[k] )                       O(N)
        }
        print ( "\n" )
    }
}
```

TC : $O(N^3)$

Break 10:43

Given an array A[]. Return the length of smallest subarray which contains both max and min element of the array.

A = [0:3 1:6 2:2 3:1 4:6 5:5]

min = 1

max = 6

ans = 2

A = [2 2 6 4 5 1 5 2 6 4 1]

max = 6

min = 1

ans = 3.

Brutforce

⓪ Find max & min

→ ① Find all the subarrays → $O(N^3)$

→ ② Check each subarray for max & min → $O(N^3)$

↓

If the subarray contains max & min val then updat the ans with min length.

Valid subarray.

max ... min

↓

max or min

max min min

The subarray will either start or end at min/max

max min

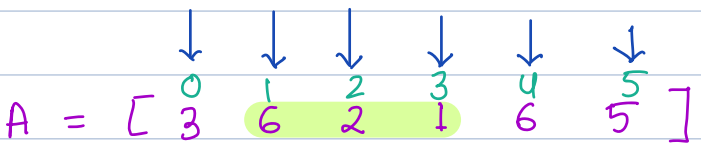
min max

} consider both of these cases.

max ④ max ② max ③ min ①

max min

For all min values find the closest max on the left.



Case (i) [max min]

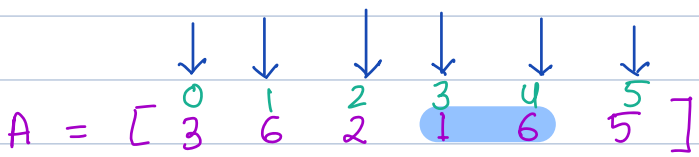
mxval = 6
mnval = 1

mxidx = ~~1~~ 4

length = i − mxidx + 1

= 3 − 1 + 1

= 3



Case (ii) [min max]

mxval = 6
mnval = 1

mnidx = 3

length = i − mnidx + 1

= 4 − 3 + 1 = 2

Pseudocode

```
// Step 1   calculate   min   and   max   values
    mnval          // store   minval
    mxval          // store   maxval
    minlength  =   ∞     // INT_MAX
// case (1)   [max   min]
    mxidx  =  -1                                   → O(N)
    for ( i ⟶ 0 to N-1 ) {
        if ( A[i] == mxval ) {
            mxidx = i
        }
        if ( A[i] == mnval && mxidx != -1 ) {
            length = i - mxidx +1
            minlength = min (minlength , length)
        }
    }
```

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

```
// case (2)   [min   max]
    mnidx  =  -1                                   → O(N)
    for ( i ⟶ 0 to N-1 ) {
        if ( A[i] == mnval ) {
            mnidx = i
        }
        if ( A[i] == mxval && mnidx != -1 ) {
            length = i - mnidx +1
            minlength = min (minlength , length)
        }
    }
```

}

print (minlength)

TC : O(N)
SC : O(1)

Doubt Senion

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

A = [1 2 3]

f i ⟶ start

f j ⟶ end

for k ⟶ i to j

integer ⟶ -2^{32} ⟶ $2^{32}-1$

$10^3 * 10^9$

10^{12}

n = 1000

A[i] = 10^9

-10^{18} to 10^{18}

⟹ long