Arrays - Carry forward & Subarrays

Contest after module

1.5 hrs → 3 questions

Question

Given string s of lowercase chars, return count of pairs (i,j) s.t. i<j && s[i] = 'a' && s[j] = 'g'.

eg s = a b c g a g (0 1 2 3 4 5)

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

(2,3) (2,4) (2,7)

(5,7) (6,7) ans = 5

for every 'a', we need total 'g' (s) in the right.

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

TC: $O(N^2)$

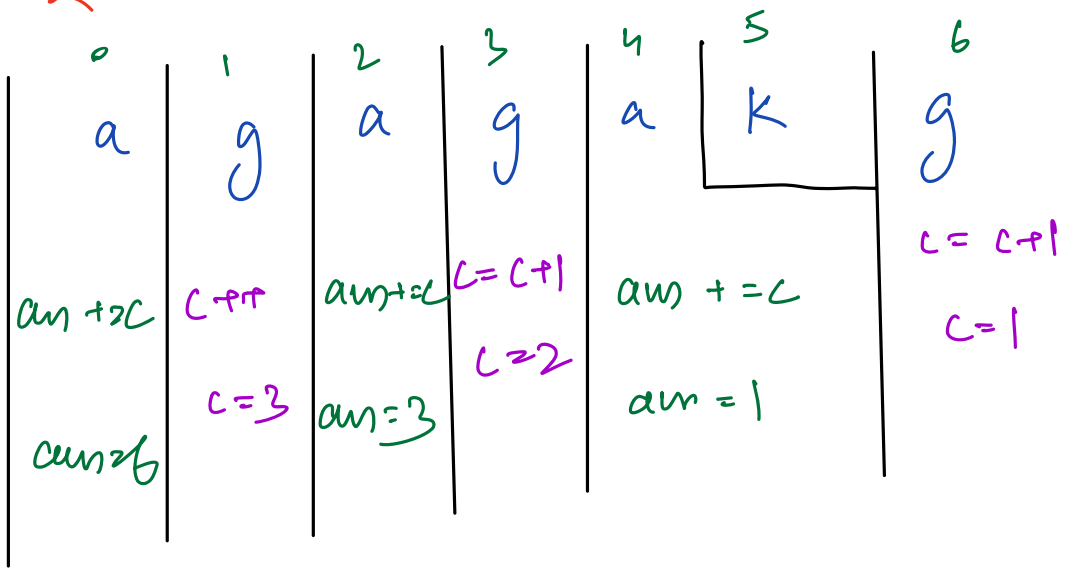
SC: $O(1)$

a g a g a k g

c = 0

no. of g(s) in right side

ans = 0



```
ans = 0 , c = 0

for ( i = n-1 ;  i >= 0 ; --i ) {
    if ( s[i] == 'g' ) {
        ++c ;
    }
    if ( s[i] == 'a' ) {
        ans += c
    }
}
print(ans)
```

TC : O(N)

SC : O(1)

TODO:

traverse from left to right

Subarray

- → continous part of array
- → sigle element / complete array
- → empty array **is not** subarray
- → i ..... j length : j-i+1

start end

4 1 2 3 -1 6 9 8 12

2 3 -1 6 ✓

4 12 ✗

1 2 6 ✗

6 9 8 ✓

2 4 1 6 -3 7 8 4

{1,6,8}

{1,4}

{6,1,4,2}

{7,8,4} ✓

0 1 2 3 4 5 6

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

↑ i=0

index 0

j=0 [4]

j=1 [4,2]

j=2 [4,2,10]

[4,2,10,3]

j=6 [4,2,10,3,12,-2,15]

am=7

0 1 2 3 4 5 6

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

i=1

index |

~~j=0~~

j=1 [2]

j=2 [2, 10]

⋮

j=6 [2, 10, 3, 12, -2, 15]

am = 6

A[4] = 2 6 3 9 (0 1 2 3)

[0,0] [1,1] [2,2] [3,3]

[0,1] [1,2] [2,3] 1

[0,2] [1,3] 2

[0,3] 3

4

⇒ 4+3+2+1 = 10

a[n] = a_0 a_1 a_2 . . - - -. a_{n-2} a_{n-1}

[0,0]

[0,1] [1,1]

[0,2] [1,2]

. .

. .

. .

[0, n-1] [1, n-1]

n n-1

[n-2, n-2] [n-1, n-1]

[n-2, n-1] 1

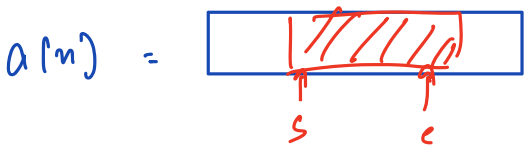
2

$$n + n-1 + n-2 + \ldots + 2 + 1$$

$$= \frac{n(n+1)}{2}$$

Question Given a[N], s & e integers.

Print subarray from [s, e] $s <= e$



2 4 5 9 8

s=1 e=3

4 5 9

```
for (i=s; i<=e; ++i) {
    print(a[i])
}
```

Question

Given a[N], print all possible subarrays.

a[] = [1, 2, 3]

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

total subarrays = $\frac{n(n+1)}{2}$

$\approx O(N^2)$

worst-case time to print subarray → O(N)

$\Rightarrow O(N^2) \times O(N)$

$= O(N^3)$

```
void printSubarrays (a[], n) {
    for (i=0; i<n; ++i) {
        for (j=i; j<n; ++j) {
            // I have subarray [i,j]
            for (k=i; k<=j; ++k) {
                print (a[k])
            }
            print (newline)
        }
    }
}
```

TC: $O(N^3)$

SC: $O(1)$

BREAK: 10:02 - 10:12

Question

Given a[N], return length of smallest subarray which contain both maximum & minimum ele. of array

a = 1 2 6 5 4 9 5 8 1 2

min = 1

max = 9

ans = 4

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

min = 1

max = 6

ans = 3

Brute force

find min & max → O(N) ans = n

```
for (i=0; i<n; ++i) {
    for (j=i; j<n; ++j) {
        mi = false, ma = false;
        for (k=i; k<=j; ++k) {
            if (a[k] == min)
                mi = true
```

TC: $O(N^3)$

SC: O(1)

```
if(a[k] == max)
    ma = true

}
if(mi && ma){
    if(am > j-i+1)        // len of subarray (i,j)
        am = j-i+1
}
}
}
```

Optimize!

find min & max → O(N) am = n

```
for(i=0; i<n; ++i) {
    mi = false, ma = false;
    for(j=i; j<n; ++j) {
        if(a[j] == min)
            mi = true;
        if(a[j] == max)
            ma = true;
        if(mi && ma){
            if(am > j-i+1)        // len of subarray (i,j)
                am = j-i+1
        }
    }
}
```

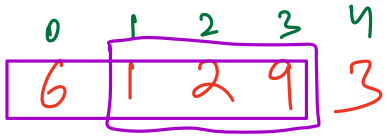
TC: $O(N^2)$

SC: $O(1)$

ans = j - i + 1

3 3 break;

3

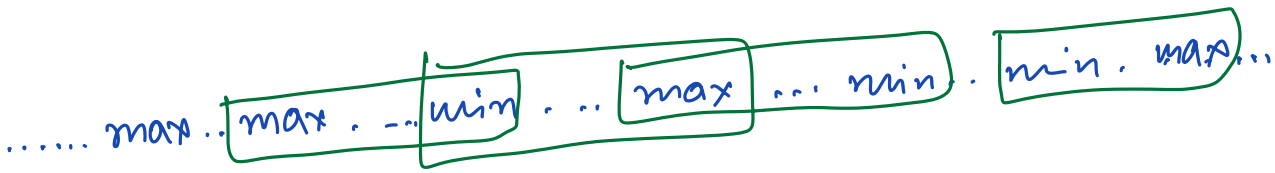


min = 1 ans = 3

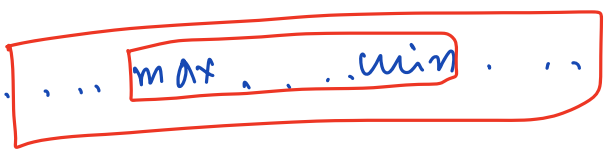
max = 9

In my final ans subarray

1. Only 1 min & 1 max will be present



2. min & max values will lies on corners.

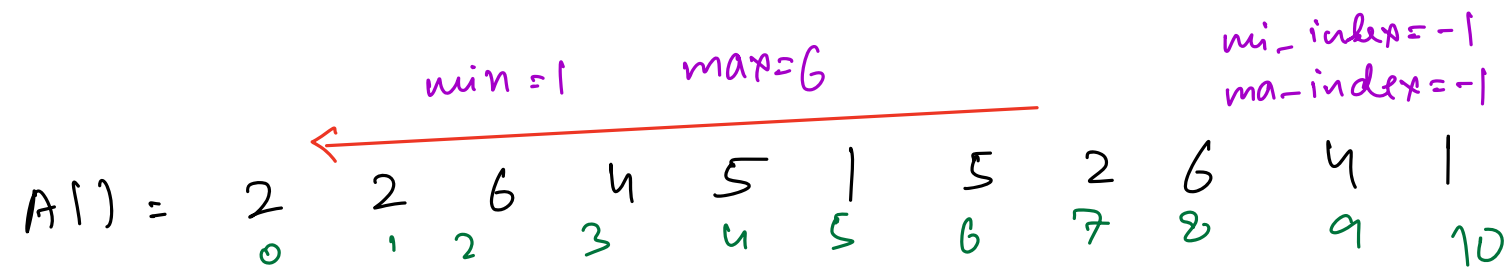


possible ans :

min ... max OR max ... min

for each min, find closest max in right

for each max, find closest min in right



0 2

1 2

2 6 — ma_index = 2, mi_index = 5 — [2, 5] len = 4

3 4

4 5

5 1 — min index = 5, ma_index = 8 — [5, 8] len = 4

6 5

7 2

8 6 — ma_indx = 8, mi_index = 10 — [8, 10] len = 3 — am = 3

9 4

10 1 — mi_index = 10, ma_indx = -1

mi_index = -1

ma_index = -1

ans = n

find min & max → O(N)

```
for (i=n-1 ; i>=0; --i) {
    if (a[i] == min) {
        mi_index = i
        if (ma_index != -1) {
            len = ma_index - mi_index+1
            if (len < ans)
                ans = len
        }
    }
    if (a[i] == max) {
        ma_index = i
        if (mi_index != -1) {
            len = mi_index - ma_index+1
            if (len < ans)
                ans = len
        }
    }
}
```

TC: O(N)

SC: O(1)

Doubt Session

```
for i = 0 to 2^n {
    j = i
    while (j > 0)
        j--
}
```

i = 0	j = [i, 0]
1	0
2	1 +
⋮	2 +
$2^n - 1$	⋮
	$2^n - 1$

$$\underbrace{1 + 2 + \ldots + 2^n - 1}_{2^n - 1 \text{ terms}}$$

$$\frac{(2^n - 1)(2^n)}{2} = \frac{2^n \times 2^n - 2^n}{2}$$

$$= \frac{2^{2n}}{2} \Rightarrow 2^{2n}$$

$$\approx O(2^{2n})$$

$$\Rightarrow O(4^n)$$