Arrays: Carry Forward

April 1, 2022

Agenda:

* 3 interesting problems
  using the concept of carry-forward technique

Q.1. Count pairs AG

Given a string, calculate no. of pairs (i,j) such that
$\boxed{i<j}$, s[i] = 'a' and s[j] = 'g'

i, j → characters?
i, j → indices.

e.g. b a a g c

Note: All characters are lower case only.

String → an array of characters.

arr[i]
↓
str[i] → ith character.

for ele in arr:

for ch in str:

b a a g c .

Ans = 2

Quiz.

a c g d g a g

3+1 = 4. ✓

b c a g g a a g

(a,g)

3+1+1 = 5 ✓

Brute force: → O(N²)

```
cnt = 0
n = len(s)
for i in range(0, n):            → n
    for j in range(i+1, n):
        if s[i] == 'a' && s[j] == 'g':
            cnt += 1

return count.
```

n-1, n-2, n-3, ..., 1

0, n

1, n

i, n

i+1, n

O(n²)

```
cnt = 0
n = len(s)
for i in range(0, n):            → n
    if s[i] == 'a':
```

```
for j in range(i+1, n):
    if s[j] == 'g':
        cnt += 1
```

$O(N^2)$

Can we bring this to $O(N)$?

e.g.

a d g a g a g f g

cnt = ?

* For every a, count no. of g's to the right of a.

			4	3	3	2	2	1	1

a

Prefix / Suffix array.

Think, without having to use an extra array.

TC: $O(N)$

SC: $O(1)$

a d g a g a g f g

4

3

a,

cnt g = 1

a, g

a,g a,g a,g a,g

cnt-g = 4.

a,g a,g a,g

cnt-g

2

2

1

cnt-g

ans = 0

$\boxed{\text{cnt\_g} = \cancel{0}\ 2}$

ans = 2

cnt_g = 2+1 = 3

ans = 2 + 3 = 5

ans = 5

cnt_g = 3.

cnt_g = 4
ans = 5 + 4 = 9.

$$
\begin{array}{r}
1\;1\;\; \\
3\;4\;5 \\
+\;8\;8\;7 \\
\hline
1\;2\;3\;2
\end{array}
$$

* Carry Forward Technique.

Pseudo-code.

```
n = len(s)
cnt_g = 0
ans = 0
for i in range(n-1, -1, -1):
    if s[i] == 'g':
        cnt_g += 1
    else if s[i] == 'a':
        ans += count_g
return ans.
```

range(1, N) → 1 to N-1
range(n-1, -1) → ~~1 to~~ n-1 to 0

$\boxed{\text{TC: } O(N) \quad \text{SC: } O(1).}$

** → Count no. of g's to right of 'a' ∀ 'a'

Or " a's to left of 'g' ∀ 'g'

Forward iteration.

~~start~~ reverse iteration

Q.2. Leaders in an array

Given an array A of size N, you have to count the no. of leaders in the array A.

An element is a leader if it is strictly greater than all the elements to its right side.

Note: A[n-1] is always considered a leader.

e.g. 15 -1 7 2 5 4 2 3 → $O(N^2)$

5 leaders.

Quiz. 16 17 17 4 3 5 2

strictly greater.

3 leaders.

{if(a[i] < a[i-1]):

if (a[i] > max)
max = a[i] }}.

Optimised:

15 -1 7 2 5 4 2 3

max

Pseudo code.

```
ans = 1          // [n-1] will always be a leader.
maxi = arr[n-1]

for i in range(n-2, -1, -1):

    if arr[i] > maxi :
        // arr[i] is a leader.
        { ans += 1
        { maxi = arr[i].

return ans.
```

// Carry forwarding → ?
Max no. encountered.

Break till : 22:25.

[2 3 6 8 9 10 5 3 2 6]

[

\+

→ [ ]

Subarrays.

[1, 3, 5, 7, 6, 2]

Contigous part of an array

- Single element ? ✓ — [2]
- Full array ? ✓ — arr.
- Empty array ? ✗ — [].

Slice operator

arr.

arr [2 : 5] → Subarray.

arr[2 :] → Subarray
Start 2 to end.

arr [: 3] → Subarray
Starting 0 to 2.

Privileges.

sort ()
min
max.

sort () → O(n log n)

min (array) → O(N)
min (a, b) → O(1)

Q.3: Closest Min Max

Given an array A, find the length of the smallest subarray which contains both max and min of the array.

e.g. 1 2 3 1 3 4 6 4 6 3

Ans = 4.

QUIZ TIME

Q. 2 2 6 4 5 1 5 2 6 4 1 → 3

Q. 1 6 4 2 7 7 5 3 1 1 5

~~X~~ Ans = 4

Observations:

1. In the resultant subarray, maxA and minA will appear exactly once.

   7 5 3 1 ~~1~~

2. Resultant subarray, will have boundaries as minA and maxA.

   ~~X~~ $\boxed{7\ 5\ 3\ 1}$ ~~X~~

2 2 6 4 5 1 5 2 6 4 1

max

// Max and min is known → $O(n)$ ✓

ans = 4.

Search will be uni-directional,

Pseudo-code:

Find min & max.

∀ i, if arr[i] = max
find the next available min towards right. → j

if arr (i) = min
find the next available max towards right.

i ———— j

j − i + 1 → length of the subarray

Pseudo-code

# Amin, Amax
ans = N.

N ← for i in range (0, n):
    if A[i] == Amin :
        N for j in range(i+1, n):
            if A[j] == Amax :

$O(N^3)$ ✗

j − i + 1

Amin . . . . . Amax

Min .... Max

ans = min(ans, j-i+1)
break

Max.... Min.

```
else if A[i] == Amax :
    for j in range(i+1, n) :   (N)
        if A[j] == Amin :
            ans = min(ans, j-i+1)
            break
return ans.
```

N(N+N) N^2

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

j=5.

j=X

i=0.
ans=N.

i=2.
ans = N.
ans = min(N, j-i+1)
= min(N, 5-2+1)
= 4

i=5

i=8

j=10,

ans = min(ans, j-i+1)
= 3

Optimised approach.

* carry forward min i and max-i.

(j - i + 1)

Min Max is known.

2 2 6 4 5 1 5 6 6 4 1

maxI = 2
minI = 5

4X

minI = 5
maxI = 8

ans = 4

minI = 10
maxI = 7

minI = 10
maxI = 8

ans = 3

3

ans = 4

minI
maxI

Closest index of min to the right.
closest index of max to the right.

ans = min(ans, ~~max~~ |minI - maxI| + 1)

ans = 3

Pseudo-code:

N = len(A)
Amin = min(A)
Amax = max(A)
min_i = -1 (None)
max_i = -1 (None)
if Amin == Amax: return 1
for i in range(N-1, -1, -1):
    if A[i] == Amin :

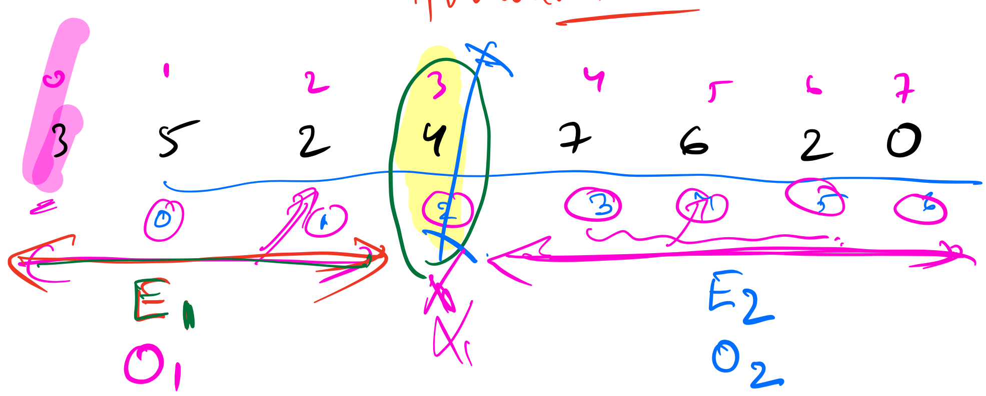
```
minI = i
if maxI != None:
    ans = min(ans, maxI - minI + 1)

else if A[i] == Amax:
    maxI = i
    if minI != None:
        ans = min(ans, minI - maxI + 1)

return ans.
```

{8, 8, 8, 8, 8} ans = 2

min = 8
max = 8

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

HW → Implement this by traversing forward.



$$E_1 + O_2 == O_1 + E_2$$

Sum of all even indexed = Sum of all od index.

$\Rightarrow$ Sum of all $\underbrace{\text{even indexe to left}}_{E_1}$ = Sum of all odd $\underbrace{\text{i to left}}_{O_1}$

+ $\underbrace{\text{Sum of all even indexed to right}}_{E_2}$ + $\underbrace{\text{Sum - odd to right}}_{O_2}$

$$E_1 + O_2 = E_2 + O_1$$