Arrays : Carry Forward

Dec 1, 2023

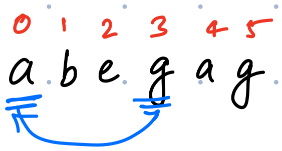
AGENDA

- Carry forward concept with & interesting questions
- Brief on subarrays

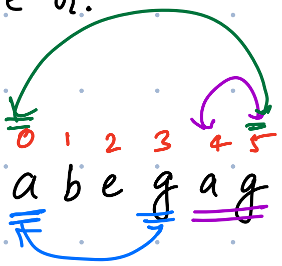
Q. Count pairs 'AG'

Given a string 's' of lowercase characters, return the count of (i,j) such that i<j and s[i]=='a' and s[j]=='g'.

e.g. →

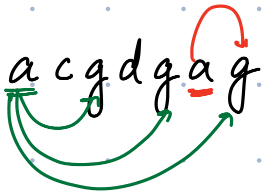


Simplify : Count no. of AG pairs such that A comes before G.

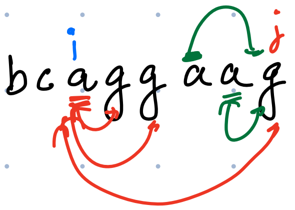


i	j		
0	3	i<j	s[i]==a s[j]==g
4	5	i<j	s[i]==a s[j]==g.
0	5	i<j	s[i]==a s[j]==g

3 pairs



3 +1= 4 pairs.



5 pairs.

:

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

i j

2,3
2,4
2,7
5,7
6,7

i<j

s[i]=a
s[j]=g. ✓

B.F

* Run a nested loop.
* For every 'a' encountered, count no. of 'g's to the right.

```
cnt=0
for(int i=0; i<n; i++)
{
        if(s[i] != 'a')
              continue;

        for( int j=i+1; j<n; j++)
        {
                if(s[j]=='g')
                       cnt++
        }
}
return cnt
```

You can also write i<n-1.

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

aaaaaaaaag.

T.C.

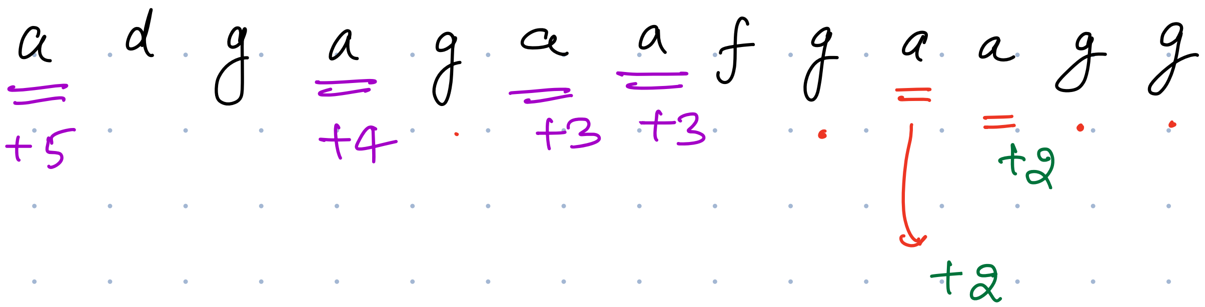
↓

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

S.C. = $O(1)$

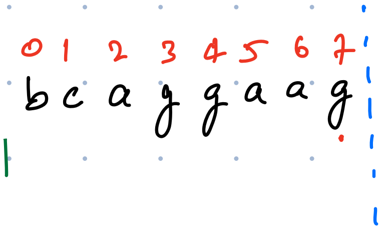
not good enough :)

Optimisation !



ans = 5+ 4 + 3 + 3 + 2 + 2

= 19



cnt_g = ~~1~~ ~~2~~ 3

ans = ~~1~~ ~~2~~ 5

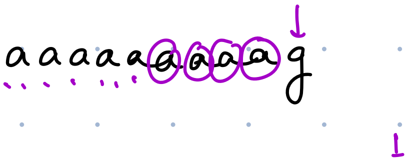
Start from end.

If you see a g,
cnt_g++

If you see a a,
ans += cnt_g

If you see another character,
ignore.

aaaaaaaaaag.



cnt = 1+1+1+1+ . . .

Code.

ga

```
ans=0, cnt_g=0
for (int i=n-1; i>=0; i--)
{
        if (s[i]=='g')
                cnt_g++
        else if (s[i]=='a')
                ans += cnt_g
        else
            ~~// Ignore.~~
}
return ans
```

Not needed. Ignore.

T.C. = O(N)

S.C. = O(1)

"Carry forward"

H.W: :) Modify the approach so that you run loop in forward direction.

Subarrays.

↓ A $\overset{\text{continuous (one}}{\text{contigous}}$ part of an array.

Original arr: $\begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 1 & \underbrace{2 \quad 3 \quad -1 \quad 6} & & & & 9 & 8 & 12 \end{array}$

* 2 3 -1 6 ✓ 2 - 5

* 8 7-7 [A single element is also a subarray]

* 4 1 2 3 -1 6 9 8 12 0-8 [Complete array is also a subarray.]

* 4 12 XX Not a subarray. (Not contigous)

* 2 3 1 XX Order is not same.

* { } XX Empty array is not a subarray.

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

Representation of a subarray

Subarray can be represented by

(start point, end point)

(2,5)

Count no. of subarrays

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

No. of subarrays starting from index 0?

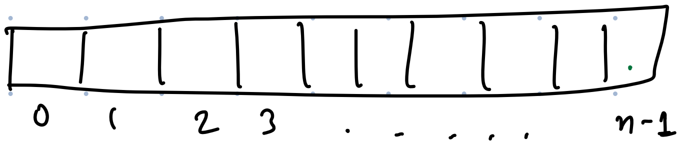
0-0
0-1
0-2
0-3
0-4
0-5
0-6

7 subarrays.

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

No. of subarrays starting from index 1 = (1,1) (1,2) (1,3) (1,4) (1,5) (1,6)

6 subarrays.



Consider all start points one by one.

No. of subarray starting from 0 = n
" 1 = n-1
2 = n-2
3 = n-3
⋮
n-1 = 1

$$1+2+3 \ldots\ldots n-1+n = \frac{n(n+1)}{2}$$ subarrays in array of size N.

Q. Print a subarray whose start point is i and end point is j.

```
for(int k=i ; k<=j ; k++)
{
        print (arr[k])
}
```

Q Print all subarrays of an array.

Generate all subarrays,

```
for(int i=0; i<n; i++)
{
        // i is fixed as a start point.

        for (int j=i ; j<n; j++)
        {
                // j is fixed as end point

                // i-j is the range of subarray,

                for (int k= i ; k<=j ; k++)
                {
                        print (arr[k])
                }
        }
}
```

Printing 1 subarray.

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

Approach:

Break till 8:30 AM

1. Generate all possible subarrays
   - Outer loop iterates over all start points
   - Inner loop " " end points.

2. Once you get the range, iterate over the range to print it

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

$$1\quad 2\quad 3\quad \underbrace{1\quad 3\quad 4\quad 6}\quad 4\quad 6\quad 3$$

max=6
min= 1

ans= 4

$$2\quad 2\quad 6\quad 4\quad 5\quad 1\quad 5\quad 2\quad \underbrace{6\quad 4\quad 1}$$

min=1
max =6

contains both 1 and 6 and is smallest.

ans=3

$$1\quad 3\quad 0\quad 4\quad 2\quad 7\quad 0\quad 8\quad 0\quad \underbrace{0\quad 9}$$

ans=2

B.F.

* Find max of the array. →O(N)
* Find min of the array. →O(N)

* Generate every subarray, →$O(N^2)$ for each subarray:
  check if both max and min is present. →O(N)
* Store the minimum length of such subarrays in an ans variable

T.C→ $O(N^3)$

$2\ 2\ 6\ 4\ 5\ 1\ 5\ 2\ \underline{6\ 4\ 1}$

Observations

1. Min and max will always ocur at the boundaries of your final ans.

   ↓

   If not at boundaries, you can always cut-short it.

   $4\ 3\ 1\ 2\ \underline{2\ 1\ 6\ 8\ 2}\ 7\ 3$

2. Min and max will always occur once in your final subarray.

2 2 6 4 6 5 1 5 2 6 4 1

↑

min = 1

max = 6

ans = ~~5~~ 3

Approach:

1. Iterate over the array,

   If you encounter "min", iterate towards the right to find max.

   If you encounter "max", iterate " " to find min.

   Otherwise ignore.

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

↓

Not good enough.

0	1	2	3	4	5	6	7	8	9	10	11	12
1	6	4	6	5	1	5	2	6	4	4	2	1

i

Amax = 6

Amin = 1

max_i = -1 [Nearest Index of max that I have seen]

min_i = -1 (" " min "]

i=12

min_i = 12

max_i = -1

i=11, 10, 9

ignore

i = 8

max_i = 8

min_i = 12

ans = 12-8+1 = 5

i = 7, 6

ignore

i=5

max_i = 8

min_i = 5

ans = 8-5+1 = 4

i = 4

ignore

i = 3

max_i = 3

min_i = 5

ans = 5-3+1

= 3

i=2

ignore

i = 1

max_i = 1

min_i = 5

↓

ans won't be updated.

i=0

min_i = 0

max_i = 1

ans = 1-0+1 = 2 ✓

Code.

// Find the min and max of the array.
// Amin and Amax.

ans = INT_MAX (+∞) — or length of the array.
min_i = -1
max_i = -1

0	1	2	3	4	5	6	7	8	9	10	11	12
1	6	4	6	5	1	5	2	6	4	4	2	1
								i				

min_i = 12
max_i = ~~-1~~ 8

```
if( Amin == Amax) return 1.

for( int i=n-1 ; i >=0 ; i--)
{
        if (arr[i] == Amin)
        {
                min_i = i
                if (max_i != -1)
                {
                        ans = min(ans, max_i - min_i + 1)
                }
        }
        else if (arr[i]== Amax)
        {
                max_i = i
                if (min_i != -1)
                {
```

Or take |max_i - min_i| absoute diff.

```
                    ans = min(ans, min_i - max_i + 1)
               }
          }
     }
     return ans.
```

{ 8, 8, 8, 8, 8, 8, 8 }

T. C. → O(N)

S. C. → O(1)