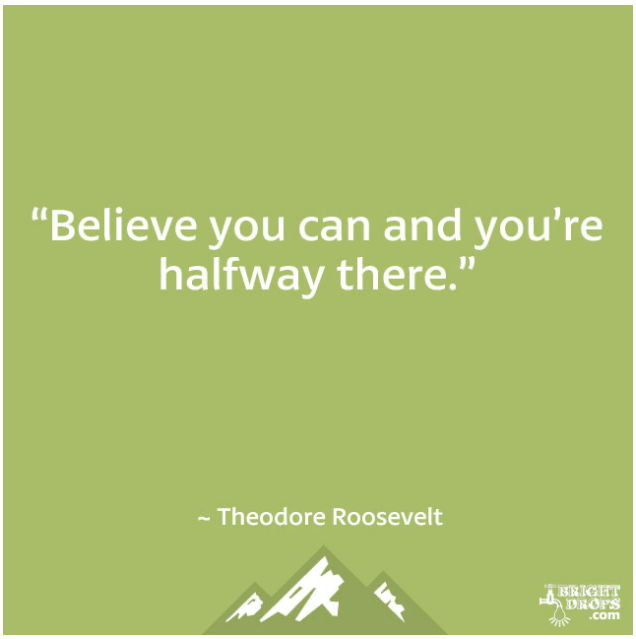
ARRAYS : CARRY FORWARD



Good Evening ☺

Today's content

01. Count pairs "ag"
02. Leaders in an array
03 Subarray basics
04. Subarray containing min & max

01. Count pairs "ag"

Given a char[]s. Calculate no. of pairs (i,j) (indices) such that $i<j$ && $s[i] = 'a'$ && $s[j] = 'g'$.

Note :- All characters are in lower case.

Constraints : $1 \le N \le 10^5$ N = len of arr

$'a' \le s[i] \le 'z'$

Eg :- s[8] : { b a a g d c a g }
0 1 2 3 4 5 6 7

Pairs (1,3) (1,7)
(2,3) (2,7)
(6,7)

ans = 5

Eg :- { b c a g g a a g }
0 1 2 3 4 5 6 7

Pairs { (2,3) (2,4) (2,7)
(5,7) (6,7) }

Ans = 5

Eg:- { a c g d g a g }
0 1 2 3 4 5 6

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

Ans = 4

Brute force Approach

↳ Check for all the pairs & count the no. of "ag" pairs

count = 0;

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

TC : $O(n^2)$
SC : $O(1)$

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

i ↓ (index 1)
j ↑ (index 2)

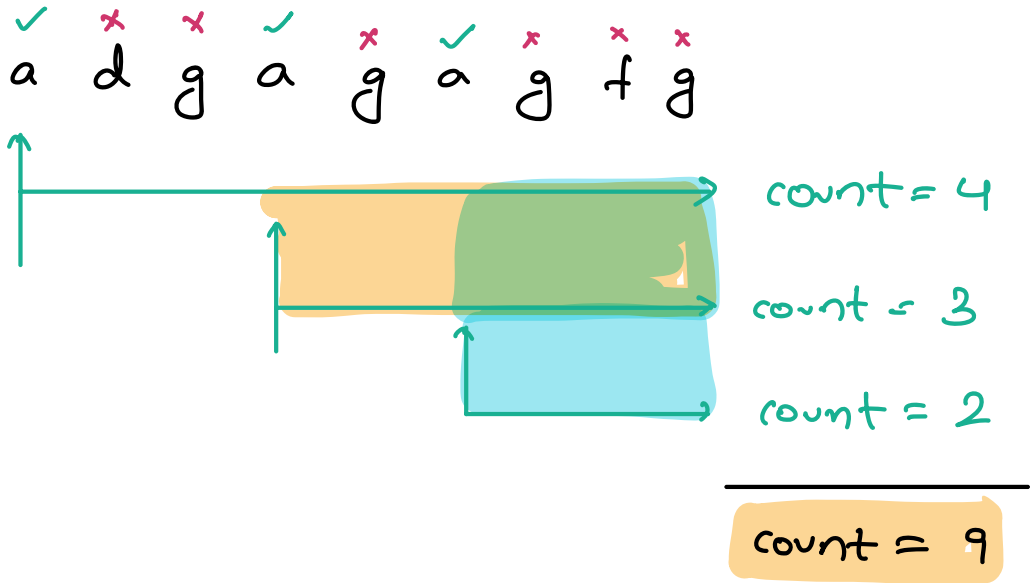
Idea 2 → Consider only correct pairs

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

TC : $O(n^2)$

SC : $O(1)$

Ex:



Optimised Approach

Count the no. of 'g' from right hand side & update the answer as soon as you have 'a'

ans = 0 (overall ans)

count = 0 (count of g)

a	d	g	a	g	a	g	f	g
ans += c ans = 5+4 = 9	✗	c = c+1 c = 4	ans = ans + c ans = 2+3 = 5	c = c+1 c = 3	ans = ans+c ans = 2	c = c+1 c = 2		c = c+1 c = 1

Ans = 9

⇒

```
c = 0 , ans = 0

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

TC = O(n)

SC = O(1)

TODO → Figure out the way to build our answer by counting no. of 'a's & update the answer at g.

Leaders in Array

Given an array arr[N], count the no. of leaders.
An ele is a leader if it is strictly greater than all the ele on right hand side

Note :- arr[N-1] is always a leader

Eg :- { 15 -1 7 4 2 5 2 3 } Ans = 4

Eg :- { 10 7 9 3 2 4 }
0 1 2 3 4 5
Ans = 3

Eg :- { 8 8 8 } Ans = 1

Brute force → For every element, iterate on to RHS & find max if (arr[i] > max) leader ++;

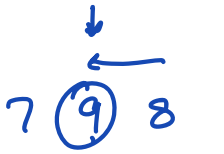
TC = $O(n^2)$
SC = $O(1)$

Optimised Idea → Carry forward the max of elements of RHS

→

```
count = 1
max = ar[n-1];
for (i = n-2 ; i ≥ 0 ; i--){
    if (ar[i] > max){
        count = count + 1;
        max = ar[i]
    }
}
```

3 3



max = 8

TC : O(n)

SC : O(1)

Subarray → Continous part of the array

Note :-
01. A single ele
02. A complete array is also a subarray
03 [ ] → No

Eg :- { 1 , 2, 3, 4, 10, 7, 6 }
0 1 2 3 4 5 6

indices → (1, 4) ⟶ {2, 3, 4, 10}

indices → (2 5) ⟶ {3, 4, 10, 7}

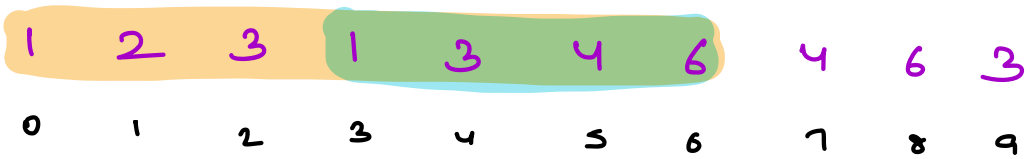
indices → (6 6) ⟶ {6}

indices → (s e) ⟶ len of subarr = e-s+1

03. Closest Min - Max

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

Eg :-



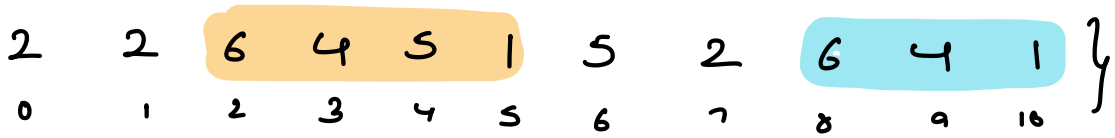
Min = 1

max = 6

subarray [0 6] → len = 6

subarray [3 6] → len = 6-3+1 = 4

Eg :- { 2 2 6 4 5 1 5 2 6 4 1 }



min = 1

max = 6

subarr = [2 5] → len = 4

subarr = [8 10] → len = 10-8+1 = 3

Eg :- {8, 8, 8, 8, 8}

min = 8

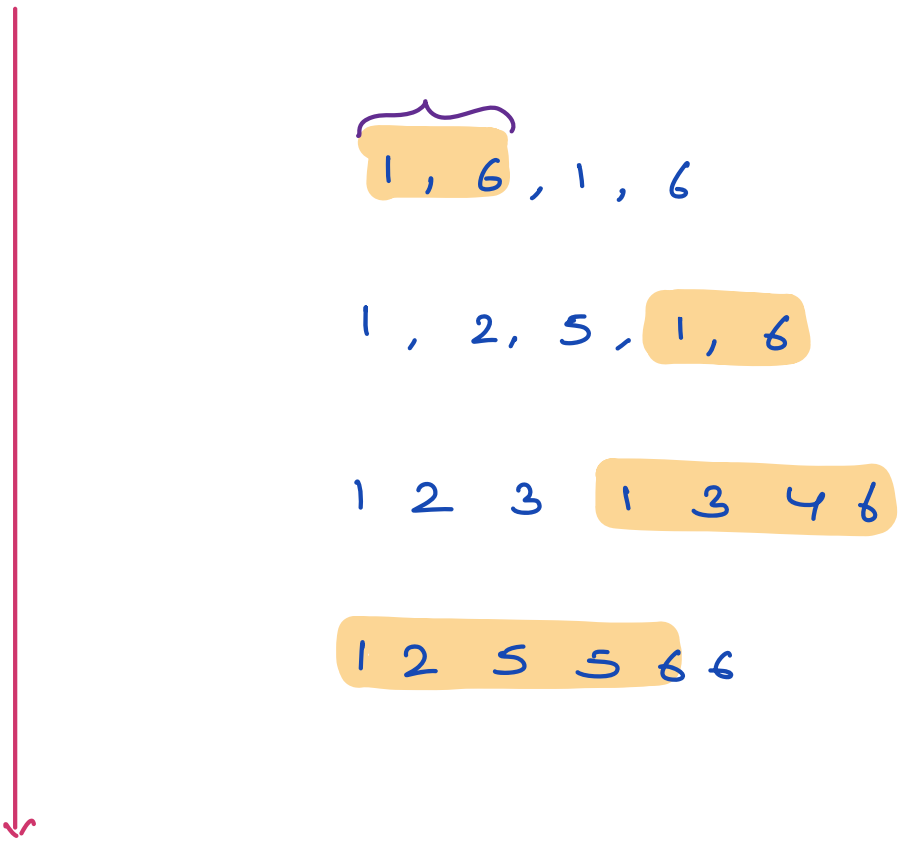
max = 8

subarray → [0 0] → len = 1

Observations

01. If min & max are same, ans = 1

02. Can you have more than one min or max in subarr



To get the smallest subarr, we only need one min & one max

03. Position of min max → should be at the corners of subarray

subarr → { .... min ... max ... }

subarr → { ... max ... min }

04. subarr [i j] ⇒ len = j - i + 1

05. Case I → {min max} if (arr[i] == min)

↓

Need to look for closest max on RHS

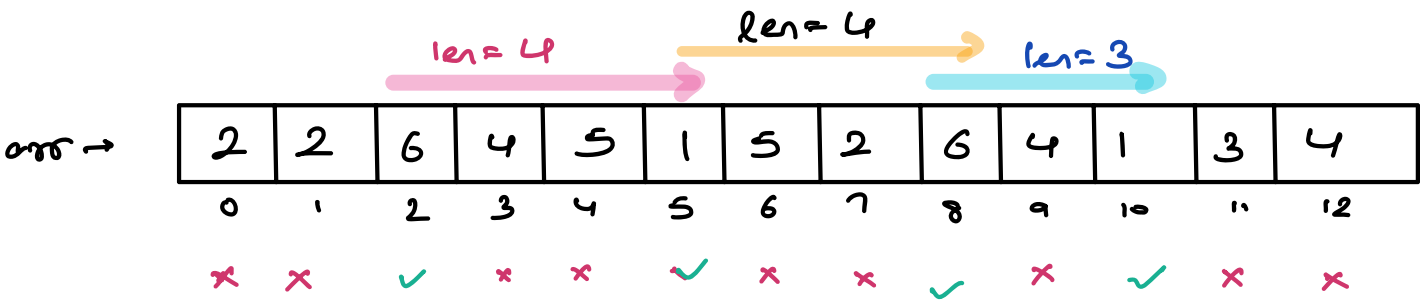
Case II → {max min} if (arr[i] == max)

↓

closest min on RHS

Min = 1

Max = 6



Ans = 3

01. Iterate & find the min & max of the array.

if (min == max) return 1

ans = n ;

```
for ( i=0 ; i<n ; i++ )
{
    if (arr[i] == min)        → look for first max
    {
        for ( j= i+1 ; j<n ; j++ ){
            if ( ar[j] == max ){
                ans= min ( ans, j-i+1 );
                break;
            }
        }
    }

    if (arr[i] == max )
    {
        for ( j=i+1 ; j<n ; j++ )
        {
            if ( ar[j] == min )
            {
                ans= min (ans, j-i+1 );
                break;
            }
        }
    }
}
return ans;
```

TC = $O(n^2)$

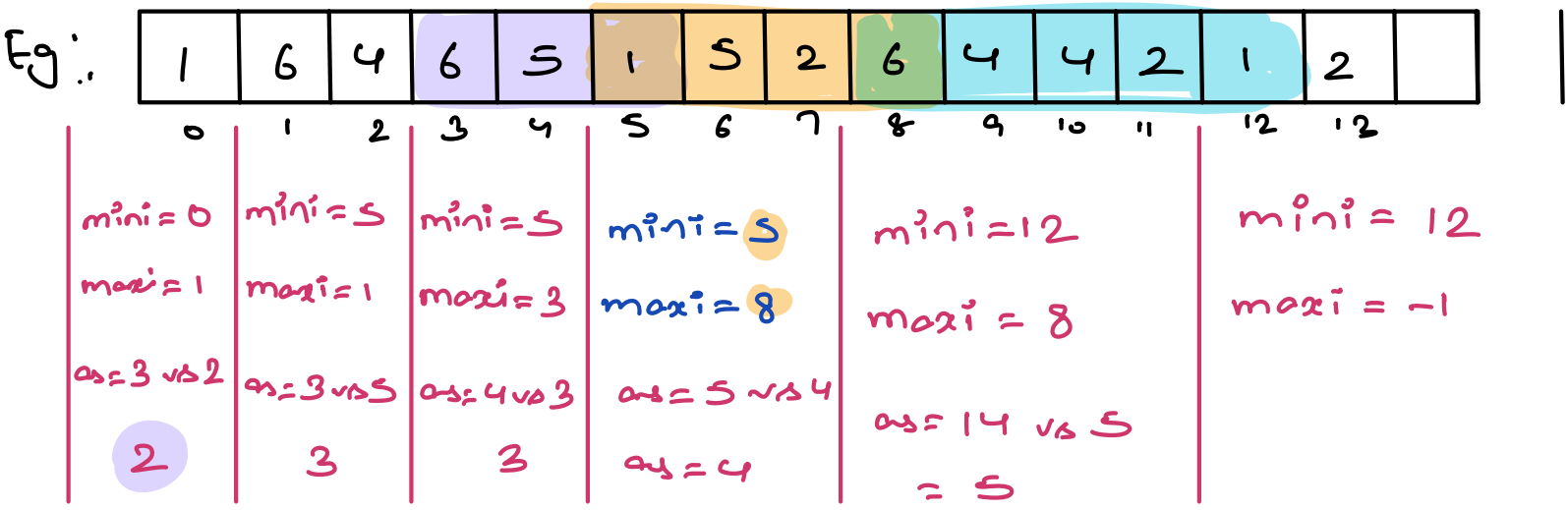
SC = $O(1)$

Idea → Carry forward the mini & maxi

Min = 1
max = 6

mini = -1
maxi = -1

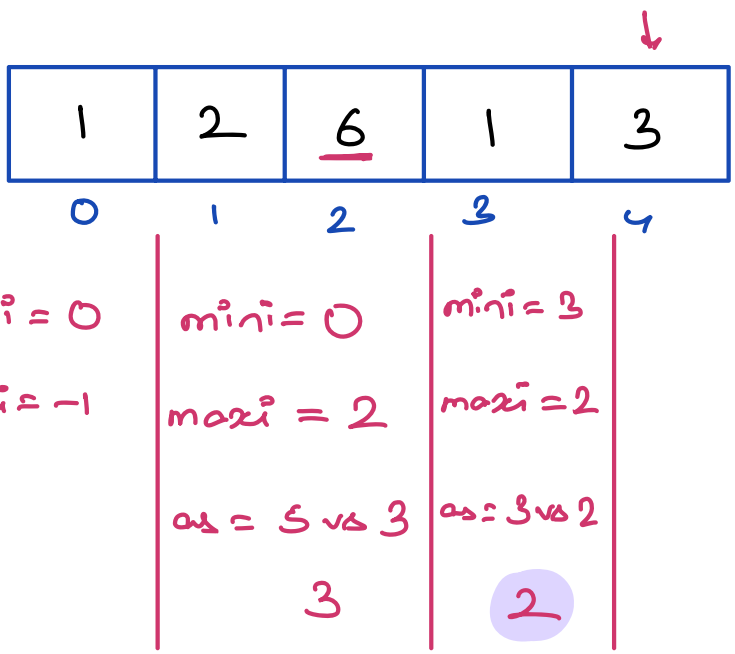
ans = 4



Min = 1
Max = 6

mini = -1
maxi = -1

ans = 5



01. Iterate & find min & max

```
if ( min == max ) return 1;
  ans=n , mini = -1 , maxi = -1
for ( i = n-1 ; i ≥ 0 ; i--){
    if ( arr[i] == min)
    {
        mini = i ;
        if ( maxi != -1 ){
            ans = minimum ( ans, maxi - mini + 1 )
        }
    }

    if (arr[i] == max )
    {
        maxi = i
        if ( mini != -1 )
        {
            ans = minimum ( ans, mini - maxi + 1 )
        }
    }
}
return ans ;
```

TC : O(n)

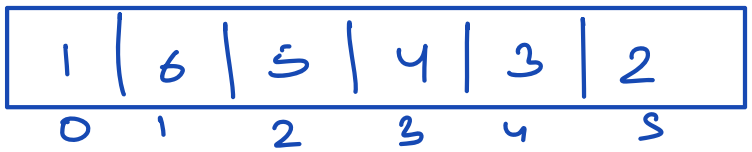
SC : O(1)

Doubt session

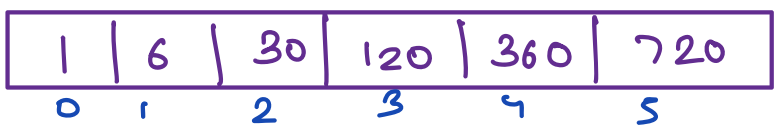
$$^{n}C_{r} = \frac{n!}{(n-r)! * r!}$$

$$^{3}C_{2} = \frac{3!}{(3-2)! * 2!} = \frac{3 * 2 * 1}{1 * 2 * 1} = 3$$

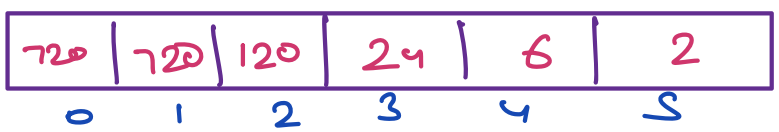
Product Array puzzle



pmul =



smul =



For a particular ar[i]

ans = pmul[i-1] * smul[i+1]