$$Ps[i] = Ps[i-1] + A[i]$$

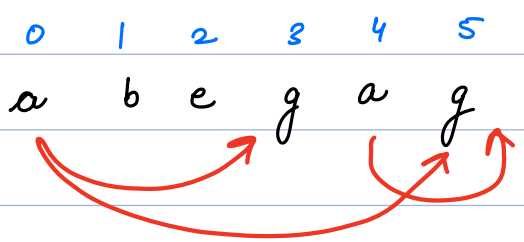
$$sum(x \to y) = pf[y] - pf[x-1]$$

- given a string of lowercase English alphabets.

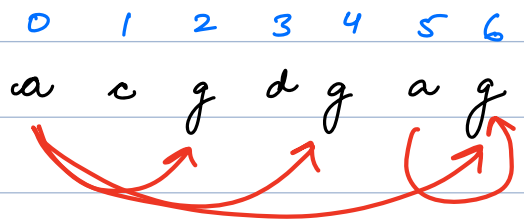
Find count of pairs (i, j) (indexes) such that

$i < j$ && $arr[i] = 'a'$

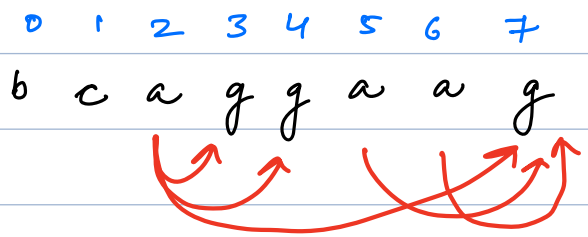
$arr[j] = 'g'$



{ 0, 3 }
{ 0, 5 }
{ 4, 5 }



ans = 4



ans = 5

Brute force → consider all pairs

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

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

for every 'a' → looking for count of 'g's on right

Traverse from end ←

OR

for every 'g' → calculate no of 'a's on left.

	b	c	a	g	g	a	a	g
cnt = 0	0	0	1	1	1	2	3	3
ans = 0	0	0	0	1	2	2	2	5

```
ans=0, cnt=0;
for( i=0; i<N; i++)
{   if( arr[i]=='a') cnt++;
    else if( arr[i]=='g') ans = ans + cnt;
}
```

T.C: $O(N)$

S.C: $O(1)$

Q Given an array of size N. Count the number of leaders in the array.

Leader: An element that is strictly greater than all the elements on its right side.

$$A[i] > A[i+1 \ldots n-1]$$

leader

Last element is always a leader.

Ex1:

0	1	2	3	4	5
10	1	3	7	4	5
	X	X		X	

Ex2:

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

ans = 3

Ex3:

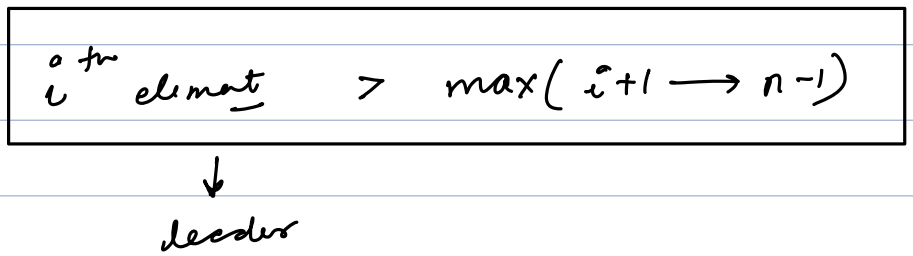
0	1	2	3	4	5	6
8	-2	4	7	6	5	1

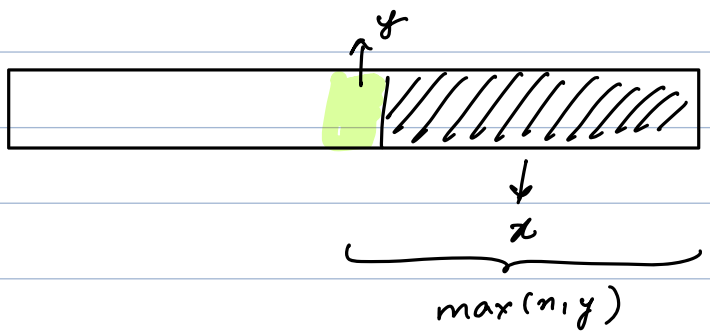
ans = 5

B-F For every index i, traverse on right

```
for(i=0; i<N; i++)
{       bool flag = true;
    for( j=i+1; j<N; j++)
    {
        if( A[j] >= A[i])
        {   flag = false; break;
        }
    }
    if ( flag) { ans++; }
}
```

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





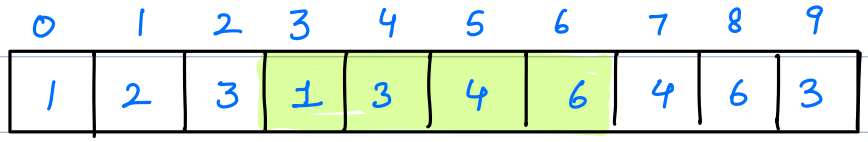
0	1	2	3	4	5	
10	7	9	3	2	4	
3	2	2	1	1	1	ans
10	9	9	4	4	4	max

```
int ans = 1;
int maxe = arr[N-1];

for( i = N-2; i >= 0; i--)
{
    if( arr[i] > mxe)
    {
        ans ++;
        mxe = arr[i];
    }
}
```

T.C: O(N)

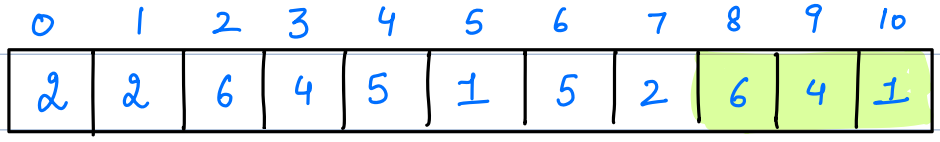
given an array of size N. Find the length of shortest range which contains both minimum and maximum element of the array.



len = 4

min → 1

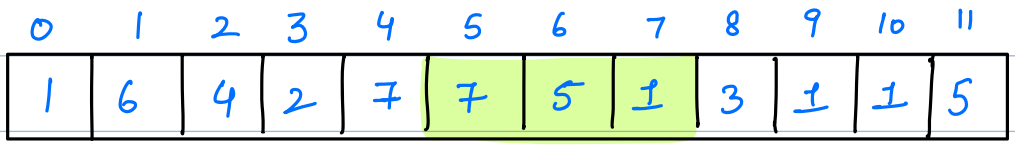
max → 6



len = 3

min → 1

max → 6

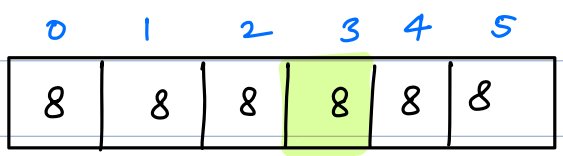


min → 1

max → 7

i → j

j - i + 1



ans = 1

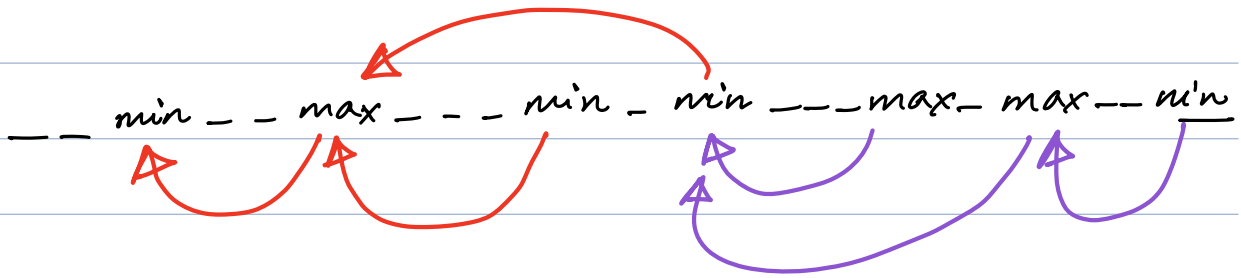
min → 8

max → 8

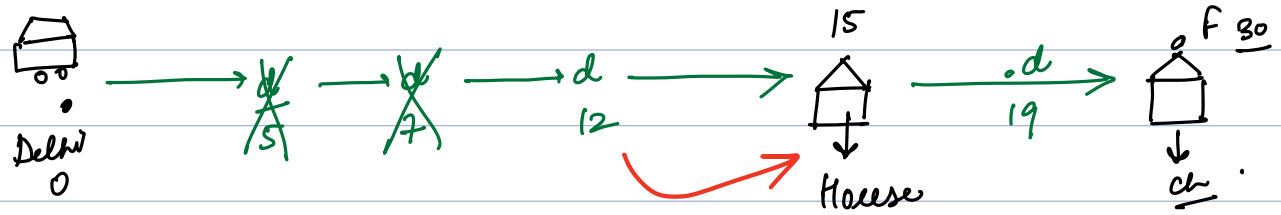
max & min → always on the border of the range

↓

Let's consider every possible pair of min & max



- For every max / min, find closest_min / closest_max on left



	0	1	2	3	4	5	6	7	8	9
	1	2	3	1	3	4	6	4	6	3
c_min = -1	0	0	0	3	3	3	3	3	3	3
c_max = -1	-1	-1	-1	-1	-1	-1	6	6	8	8

min = 1
max = 6

ans = N
~~6~~ 4

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

min = 1
max = 6

c_min = -1	-1	-1	-1	-1	-1	5	5	5	5	5	10
cmax = -1	-1	-1	2	2	2	2	2	2	8	8	8

ans = ~~11~~ ~~4~~ 3

```
// find min & max element
if( mn == mx) return 1;

cmin = -1, cmax = -1, ans = N

for( i=0; i<N; i++)
{
    if( arr[i] == mn)
    {
        c_min = i;
        if( c_max != -1)
        {
            ans = min(ans, i-c_max+1);
        }
    }
    else if( arr[i] == mx)
    {
        c_max = i;
        if( c_min != -1)
        { ans = min( ans, i-c_min+1); }
    }
}
```

T.C: O(N)
S.C: O(1)

Best Time to Buy & sell stock - I

↓

N days → stock prices

↓

allowed to do only one transaction

Buy and Sell

↓

Buy before sell

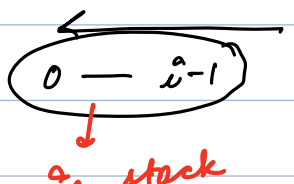
Find MAX Profit.

0	1	2	3	4	5
7	1	5	3	6	4

1) consider all pairs

/

Buy & sell



i^th

↓

selling day

→ leaders

OR

i^th

↓

buy

↓

max stock price