Question 1 :

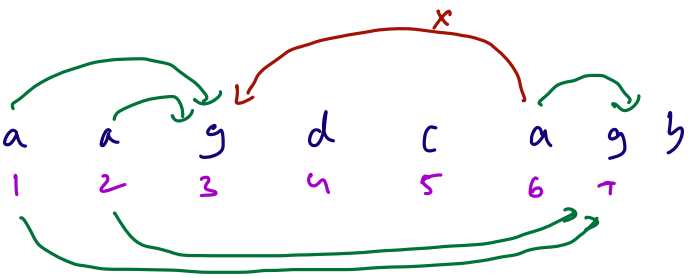
Count Pairs 'ag': Given a character array ch[N] of size N, We have to calculate the number of pairs of indices i, j where i<j && ch[i] == 'a' && ch[j] == 'g'

Constraints

- 1 <= N <= 10^5
- 'a' <= ch[i] <= 'z'

Ex1 :

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

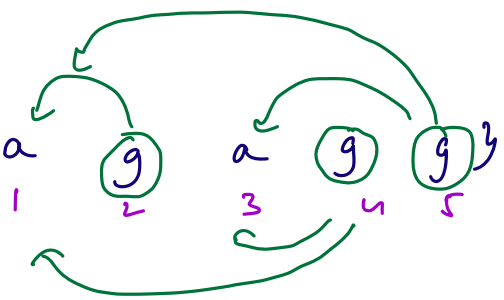


(1, 3)
(2, 3)
(1, 7)
(2, 7)
(6, 7)

Ans: 5

Quiz 1

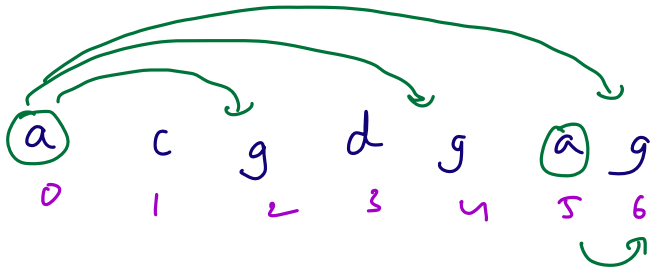
ch[6] : { b a g a g g }
0 1 2 3 4 5



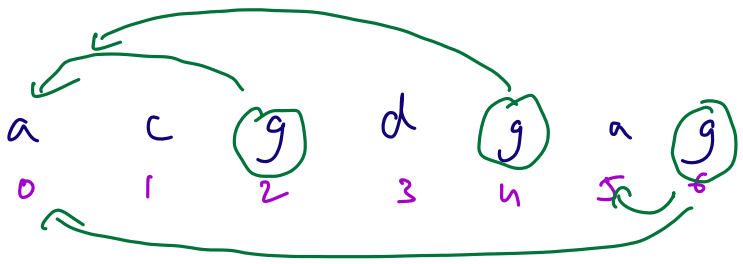
(1, 2)
(1, 4)
(3, 4)
(1, 5)
(3, 5)

Ans : 5

Quiz2:

Way1: ch[] = 

(0,2) (5,6) 4 pairs
(0,4)
(0,6)

Way2: ch[] : 

(0,2) (0,4) (0,6) 4 pairs
(5,6)

Brute Force

```
int pairs (char[] ch, int N){
    int count = 0;
    for(i=0; i<N; i++){
        for(j=i+1; j<N; j++){
            if(ch[i] == 'a' && ch[j] == 'g'){
                count++;
            }
        }
    }
    return count;
}
```

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

S.C: $O(1)$

Brute force 2

```
int pairs (char[] ch, int N){
    int count = 0;
    for(j = 0; j < N; j++){
        if( ch[j] == 'g' ){
            for(i=0; i<j; j++){
                if(ch[i] == 'a'){
                    count++;
                }
            }
        }
    }
return count;
}
```

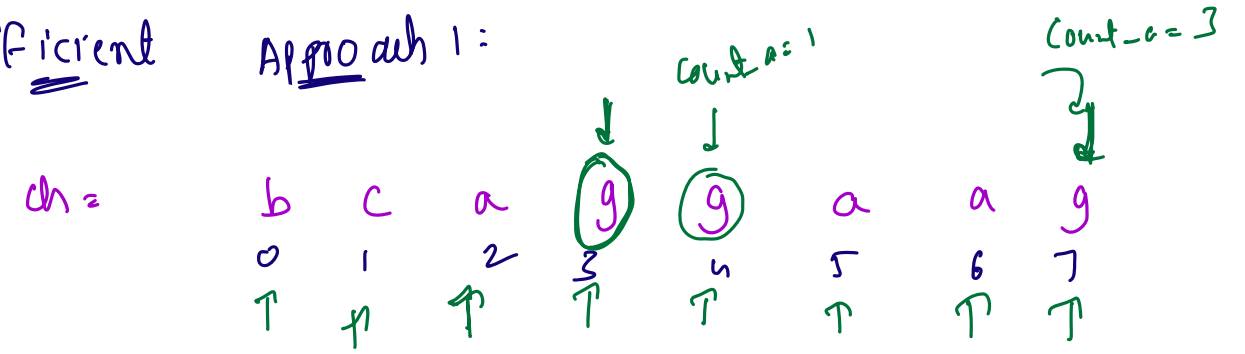
T.C:

Ex: ch[] : b h i s b z d => N like

ch[] : a g a g a g a g

1+ 3+ 5+ ..... => $O(N^2)$

Efficient Approach 1:



count_a = 0 / 1 / 2 / 3

ans = 1 + 1 + 3 = 5

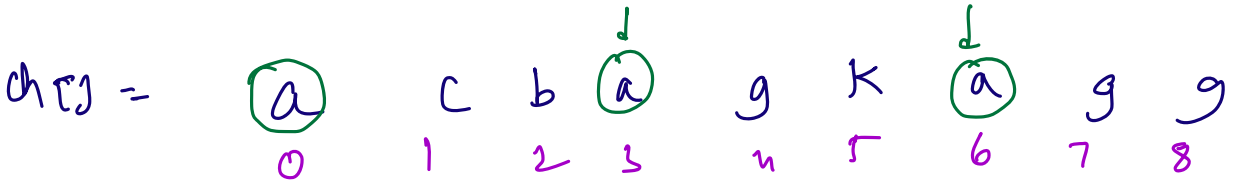
```
int pairs( char[] ch, int N){
    int count_a = 0;
    int ans = 0;

    for(i=0; i<N; i++){
        if(ch[i] == 'a'){
            count_a++;
        }
        else if(ch[i] == 'g'){
            ans = ans + count_a;
        }
    }
    return ans;
}
```

T.C : O(N)

S.C : O(1)

Efficient Approach 2:



Iterate from right to left and carry forward the no. of g's to your right

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

Question2: No. of leaders

Given an array ar[N], you have to find a number of leaders in the array.

Leader: An element ar[i] is said to be a leader if it is greater than the maximum element of all elements present on the left of it i.e [0,i-1]

Note:
ar[0] is already considered a leader.

Constraints

- `1 <= N <= 10^5`
- `1 <= ar[i] <= 10^9`

Ex:

A = 3 2 4 5 2 7 -1 15

0 1 2 3 4 5 6 7

✓ ✗ ↑ ↑ ↑ ↑ ↑ ↑

Ans = 5

Ex:

A = 4 2 3 9 7 10

↑ ↑ ↑ ↑ ↑

Ans = 3

Ex:

A = 16 17 4 3 5 2

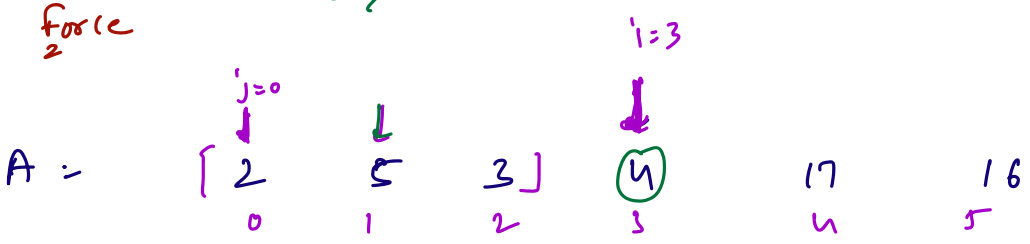
0 1 2 3 4 5

Ans = 2

Brute force

flg = True / False

count = 1

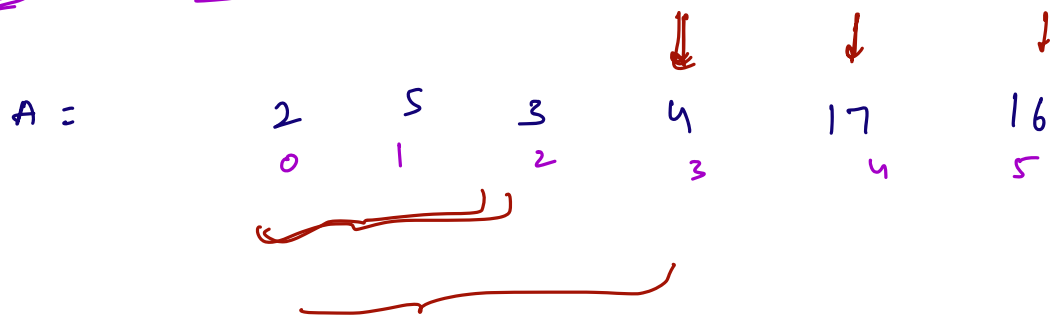


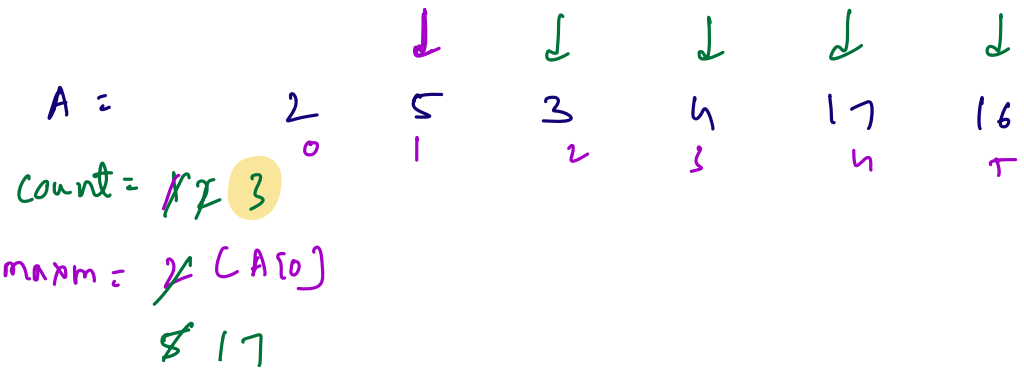

```
int leaders [int [] A , int N ){
    count = 1;
    for(i = 1; i < N; i++){
        bool flag = True;
        for(j = 0; j < i ; j++){
            if(A[j] > A[i]){
                flag = false;
            }
        }
        if(flag == True){
            count++;
        }
    }
    return count;
}
```

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

S.C: $O(1)$

Efficient Approach






```
int leaders (int[] A, int N){
    int count = 1;
    int maxm = A[0];
    for(i=1; i<N; i++){
        if ( A[i] > max){
            count++;
            max = A[i];
        }
    return count;
}
```

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

Question: Buy and Sell Stocks

Say you have an array, A, for which the ith element is the price of a given stock on day i.

If you were only permitted to complete at most one transaction (i.e, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Return the maximum possible profit.

$0 \le A.size() \le 10^5$

$1 \le A[i] \le 10^9$

N = 8

A = 3 5 3 2 4 3 8 6

0 1 2 3 4 5 6 7

Buy 0 - sell 7 => 6 - 3 = 3

Buy 2 - sell 6 => 8 - 3 = 5

Buy 3 - sell 6 => 8 - 2 = 6

A : [1, 4, 5, 2, 4]

0 1 2 3 4

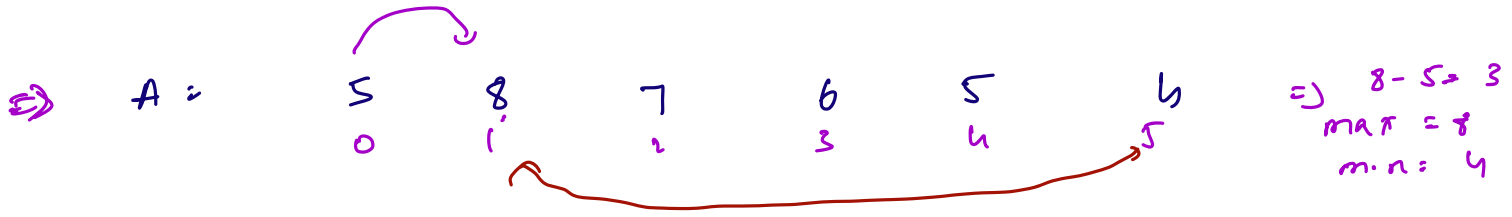
Buy 0 - sell 2 => 5 - 1 = 4

A = [6 7 5 2 6 9 3]

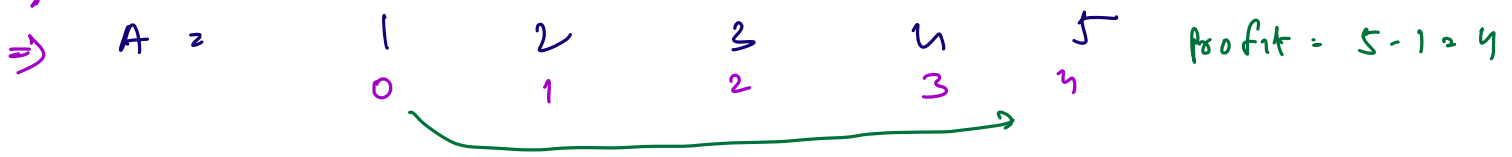
0 1 2 3 4 5 6

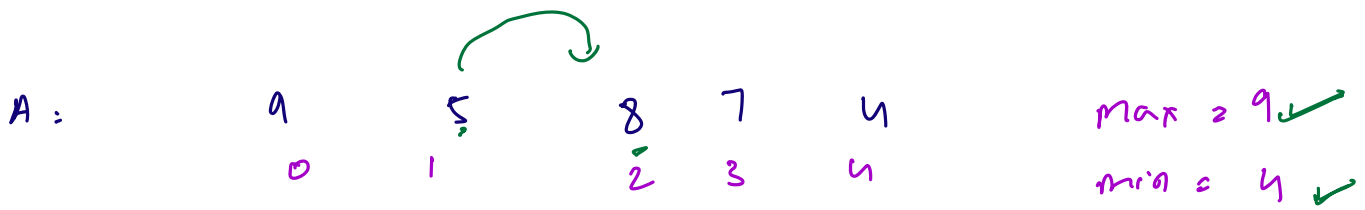
Buy 3 - Sell 5 => 9 - 2 = 7

8:34

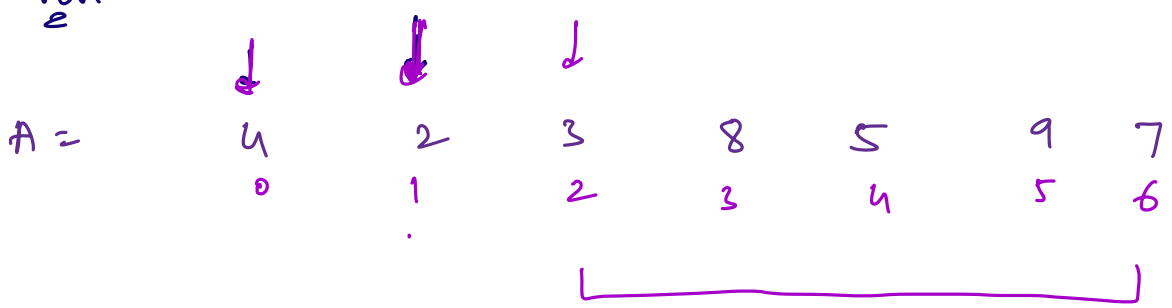


Ex:





Brute Force



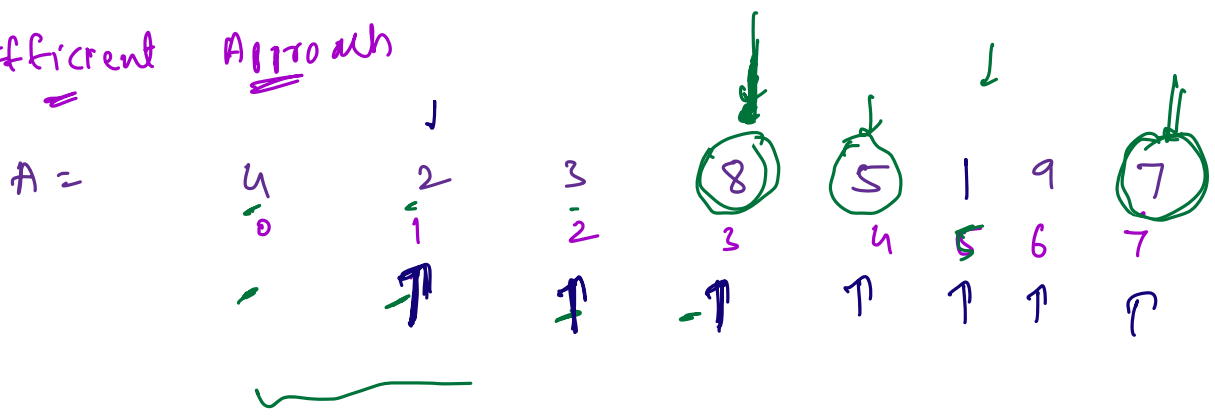

```
int maxprofit = 0;
for(i = 0; i < N; i++){
    // Buy on Day i
    for(j = i+1; j < N; j++){
        profit = A[j] - A[i];
        maxprofit = max(maxprofit, profit);
    }
}
```

return maxProfit;

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

S.C: $O(1)$

Efficient Approach



minLeft : A[0] = 4 ~~2~~ 1

maxProfit : ~~0~~ ~~4~~ ~~6~~ 8

2 - 4 = -2

3 - 2 = 1

8 - 2 = 6

5 - 2 = 3

1 - 2 = -1

9 - 1 = 8

7 - 1 = 6

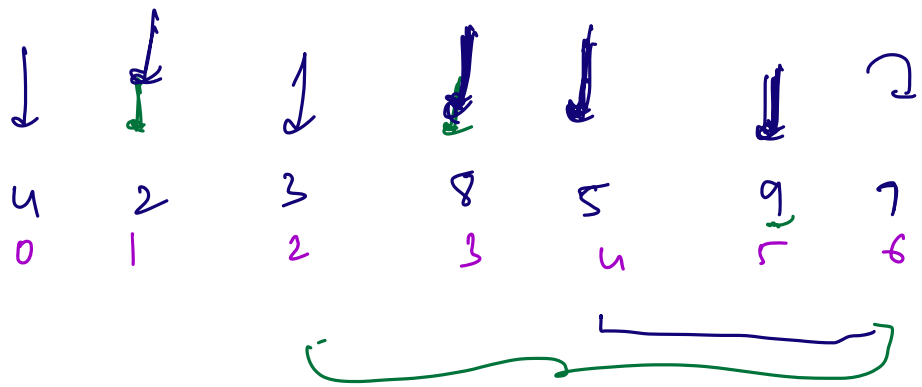
```
maxProfit = 0;
minLeft = A[0];
for(i=1; i<N; i++){
    //sell on day i
    profit = A[i] - minLeft;          // sell on day i
    maxProfit = Math.max(maxProfit, profit);
    if(A[i] < minLeft){
        minLeft = A[i];
    }
}
return maxProfit;
```

if(A[i] < minLeft)

2 3 4 5

minLeft = 2

Ways:



A = 4 2 3 8 5 9 7

0 1 2 3 4 5 6

9 - 2 = 7

maxRight = ~~7~~ 9

max_profit = ~~0~~ ; ~~4~~ ~~6~~ 7

profit = 7 - 9 = -2

9 - 5 = 4

9 - 8 = 1

9 - 3 = 6

9 - 2 = 7

9 - 4 = 5

weekend ⇒ 6-8 hrs ⇒ PSP ≈ 100% ✓

TA help Request

1 : ON
0 : OFF