Todays Content:

1. Leaders in Array
2. Buy & sell stocks
3. Ag pairs

Leaders in Array:

Given an ar[N] return no: of leaders in ar[]

An ar[i] is said to be leader if

ar[i] > max of all elements in right [i+1 .... N-1]

Note: ar[N-1] is said to be leader because no element in right

Constraints:

$1 <= N <= 10^5$

$1 <= ar[i] <= 10^9$

Ex1:

	0	1	2	3	4	5	6	7
ar[] = {	15	-1	7	2	5	4	2	3 }

ans = 5

Ex2:

	0	1	2	3	4	5
ar[] = {	10	7	9	3	2	4 }
			i			

ans = 3

Ideal: for every ar[i] Estimated TC: O(N^2)

Iterate in right [i+1 .. N-1] cal max & check if ar[i] > max : cnt

```
int leaders(int ar[], int N){ TC: O(N^2)  SC: O(1)
    int c=0;
    for(int i=0; i<N; i++){
        int max = INT_MIN;
        for(int j=i+1; j<N; j++){
            if(ar[j] > max){
                max = ar[j];
            }
        }
        if(ar[i] > max){
            cnt;
        }
    }
    return c;
}
```

TODO

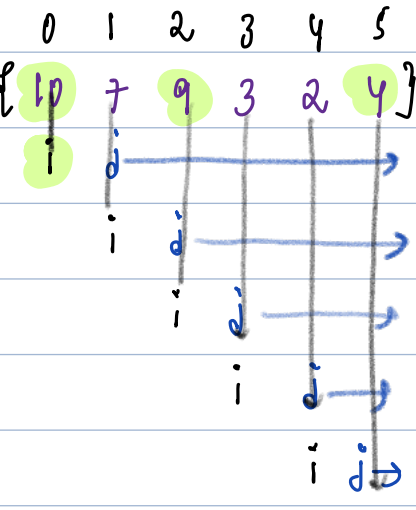
i	j

Inner loop :

Outer loop :

Total :

Tracing:



Observations:

Multiple times we are calculating max information in right

Carry forward Idea:

If Multiple times we are calculating same data from 1 direction, apply carry forward calculate data by iterating 1 time from that direction

Optimization Using Carry forward:

ar[7]:	0	1	2	3	4	5	6	
	10	7	9	3	6	4	5	max : -∞
	10 > max	7 > max	9 > max	3 > max	6 > max	4 > max	5 > max	c = 0;
return c : 4	c++		c++		c++		c++;	
	max = 10	max = 9	max = 9	max = 6	max = 6	max = 5	max = 5	

```
int leaders( int ar[]){  TC: O(N)  SC: O(1)
    int max = INT_MIN, c = 0;
    for(int i = N-1; i >= 0; i--){
        if(ar[i] > max){
            c++;
        }   max = ar[i]
    }
    return c;
}
```

Buy & Sell Stocks:

Given an ar[N], where ar[i] is price of given stock in i^{th} day

Return max profit which can be achieved by exactly 1 transaction

Note1: If we buy a stock in i^{th} day: We can sell on any day $\{i+1, i+2, i+3, \ldots N-1\}$

Note2: If cannot achieve any profit: return 0;

Constraints:

$1 <= N <= 10^5$

$1 <= ar[i] <= 10^9$

Ex1:

ar[] = { 7 1 5 3 6 4 } ans =

Ex2:

	0	1	2	3	4	5	6
ar[] = {	4	6	10	4	2	9	1 }

Idea: In stock we

	0	1	2	3	4	5	6	
i {	4	6	10	4	2	9	1 }	Profit day i^{th}

Con:

TC: SC:

Count Pairs "ab"

Given a string s calculate no: if pairs indices i, j such that

i < j && s[i] == 'a' && s[j] == 'b'

/* int N = s.size();

S = 0 1 2 ... N-2 N-1

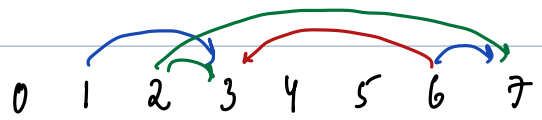
s[i];

*/

Constraints:

$1 <= N <= 10^5$

'a' <= s[i] <= 'z'



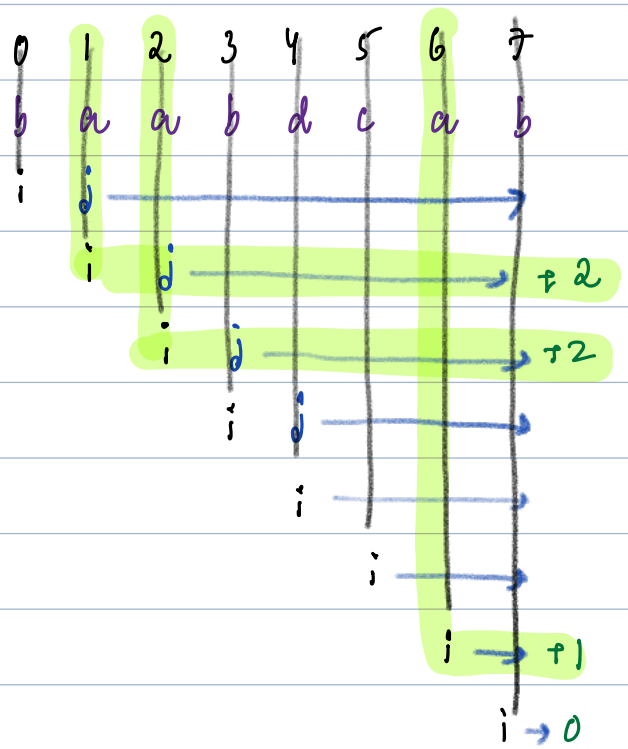
Eg1: S = b a a b d c a b

i j i j i j i j i j i<j

Pairs: (i, j) (1 3) (1 7) (2 3) (2 7) (6 7) (6 3) : 5

Ideal: For every pair (i, j): check if it forms ab & one cnt;

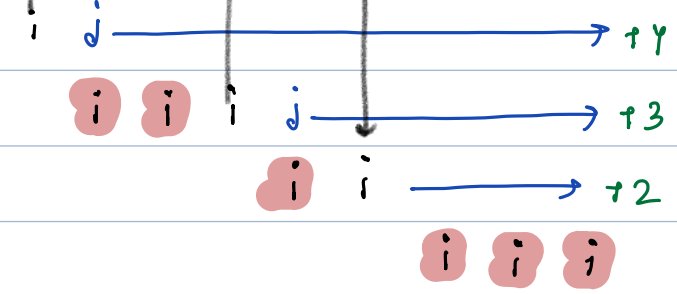
```
int pairs(String s, int N){ TC: O(N^2) SC: O(1)
    int c=0;
    for(int i=0; i<N; i++){
        for(int j=i+1; j<N; j++){
            if(s[i]=='a' && s[j]=='b'){
                c++;
            }
        }
    }
    return c;
}
```



Idea: Iterate in nested loop: if s[i] == 'a'

```
int pairs(String s, int N){
    int c=0;
    for(int i=0; i<N; i++){
        if(s[i]=='a'){
            for(int j=i+1; j<N; j++){
                if(s[j]=='b'){
                    c++;
                }
            }
        }
    }
    return c;
}
```

	0	1	2	3	4	5	6	7	8
s =	a	d	b	a	b	a	b	f	b



Obs: for every s[i] == 'a': Iterate in right & count no of b's.

Optimization Idea: Using carry forward Iterate from right to left: calculate no of b's.

	0	1	2	3	4	5	6	7	8	
s =	a	d	b	a	b	a	b	f	b	ans=0
	ans+=cb		cb++	ans+=cb	cb++	ans+=cb	cb++		cb++	cb=0
	ans=9		cb=4	ans=5	cb=3	ans=2	cb=2		cb=1	

```
int pairs(String s, int N){ TC: O(N) SC: O(1)
    int cb=0, ans=0;
    for(int i=N-1; i>=0; i--){
        if(s[i]=='b'){ cb++;}
        else if(s[i]=='a'){ ans = ans + cb}
    }
    return ans;
}
```

Idea3: a [Calculate b's on right]

obs: for every s[i] == 'b': iterate on left & count no of a's.

optimization Idea: Using carry forward iterate from left to right : calculate no: of a's.

TODO: