"Hello Everyone!"

Today's Content

1. Count Pairs
2. N Bulbs
3. Leaders in Array

(Q.1) **Count Pairs**:

→ Given a char ch[N], calculate no. of pairs

indices = i, j such that

$i < j$ and $ch[i] == 'a'$ and $ch[j] == 'g'$

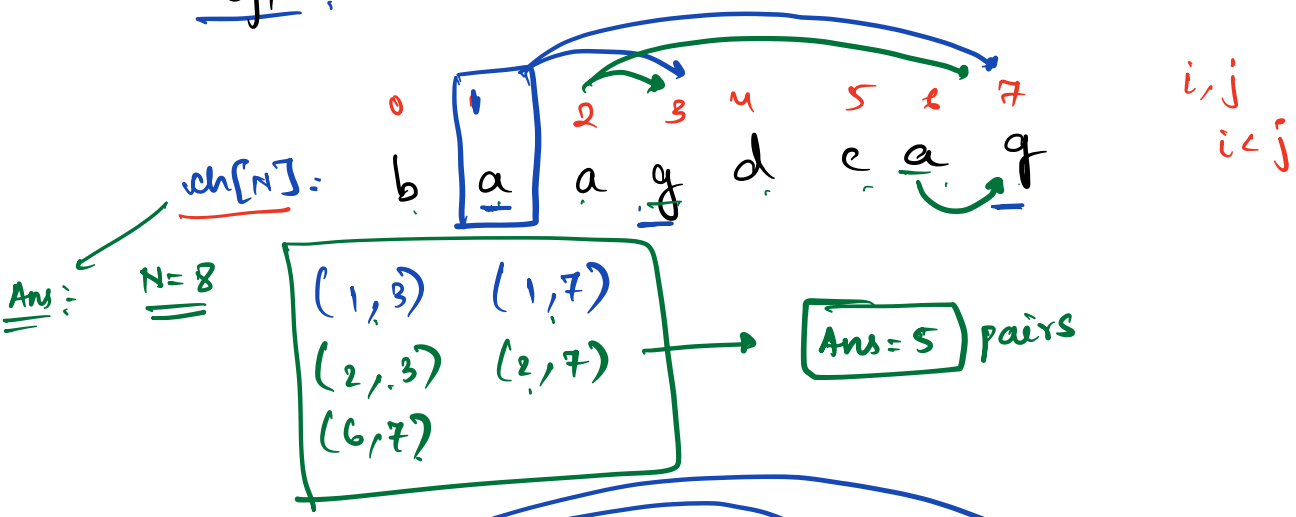
All characters are lowercase.

Constraints:

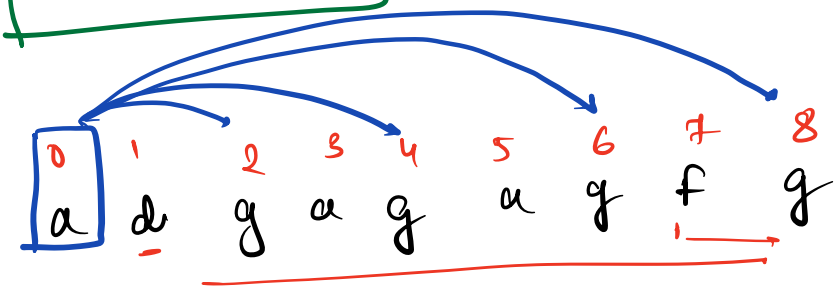
$1 <= N <= 10^5$ // N → len of array

$'a' <= ch[i] <= 'z'$

eg1 :



eg2 :



Ans :

(0, 2), (0, 4), (0, 6), (0, 8)
(3, 4) (3, 6) (3, 8)
(5, 6) (5, 8)

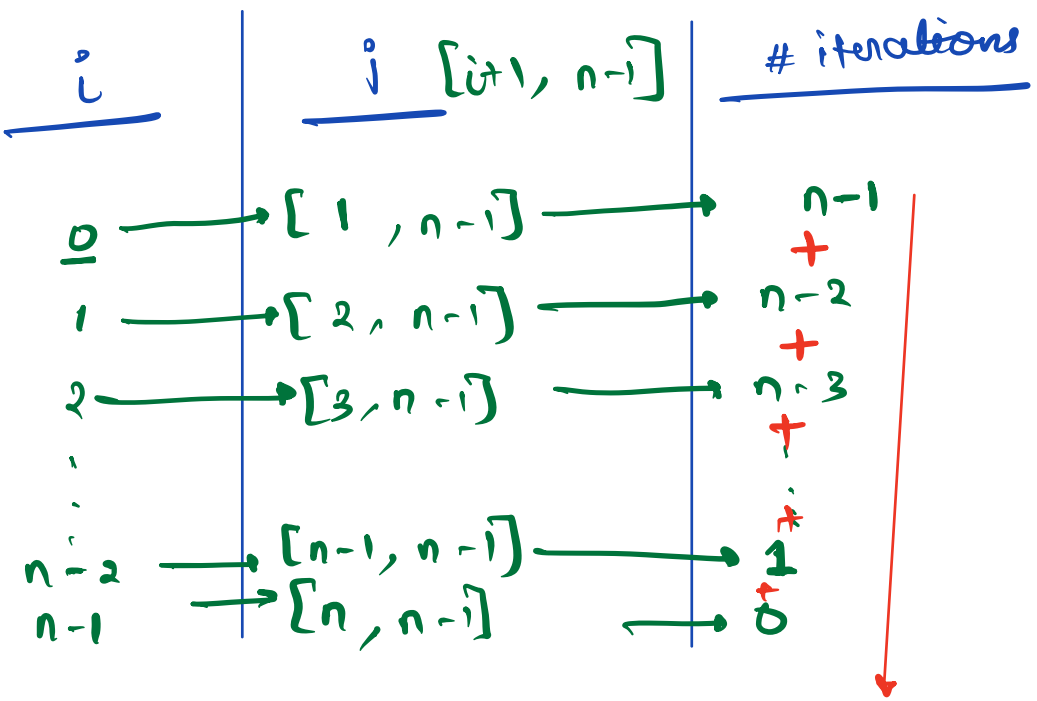
Ans = 9 pairs

Approach 1 :

```
int pairs (char ch[])
{
    n = ch.length // size of ch.
    int ans = 0
    for( i=0 ; i<n ; i++)  → [0, n-1]
    {
        // i=n-1 , j= n
        for ( j= i+1 ; j< n ; j++)
        {
            if ( ch[i]=='a' and ch[j]=='g')
            {
                ans++
            }
        }
    }
    return ans
}
```

TC : $O(n^2)$ ⇒ $N \to 10^5$, $N^2 \to 10^{10} > 10^8$ → TLE

SC : $O(1)$



iterations = $(n-1) + (n-2) + (n-3) \ldots\ldots 1 + 0$

$N \to 1$

$\frac{N(N+1)}{2}$

$= \frac{(n-1)(n-1+1)}{2} = \frac{(n-1)n}{2}$

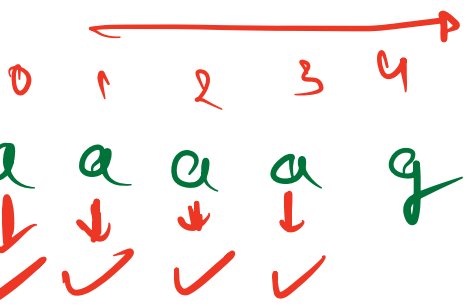
iterations = $\frac{n^2}{2} - \frac{n}{2}$

$TC \to O(n^2)$

Approach2 :-

```
int pairs (char ch[])
{
    n = ch.length // size of ch.
    int ans = 0
    for( i=0 ; i<n ; i++)  → [0, n-1]
    {
        if (ch[i] == 'a')
        {
            for ( j= i+1 ; j< n ; j++)
            {
                if ( ch[j] == 'g')
                {
                    ans++
                }
            }
        }
    }
    return ans
}
```

eg:-



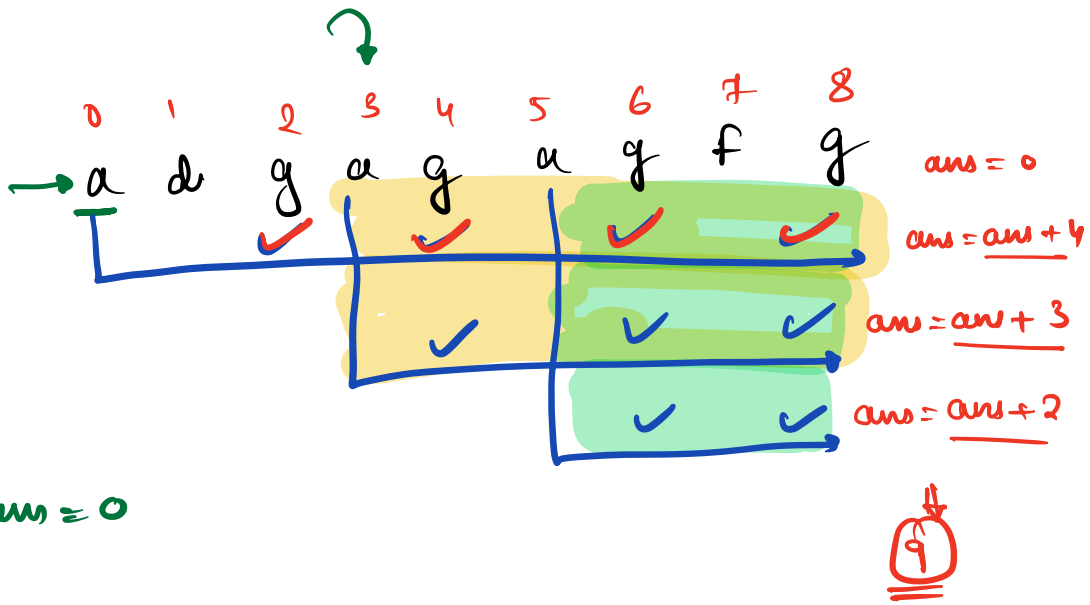
TL → $O(n^2)$ → worst case

SC → $O(1)$

eg1 :

0	1	2	3	4	5	6	7
b	a	a	g	d	e	a	g

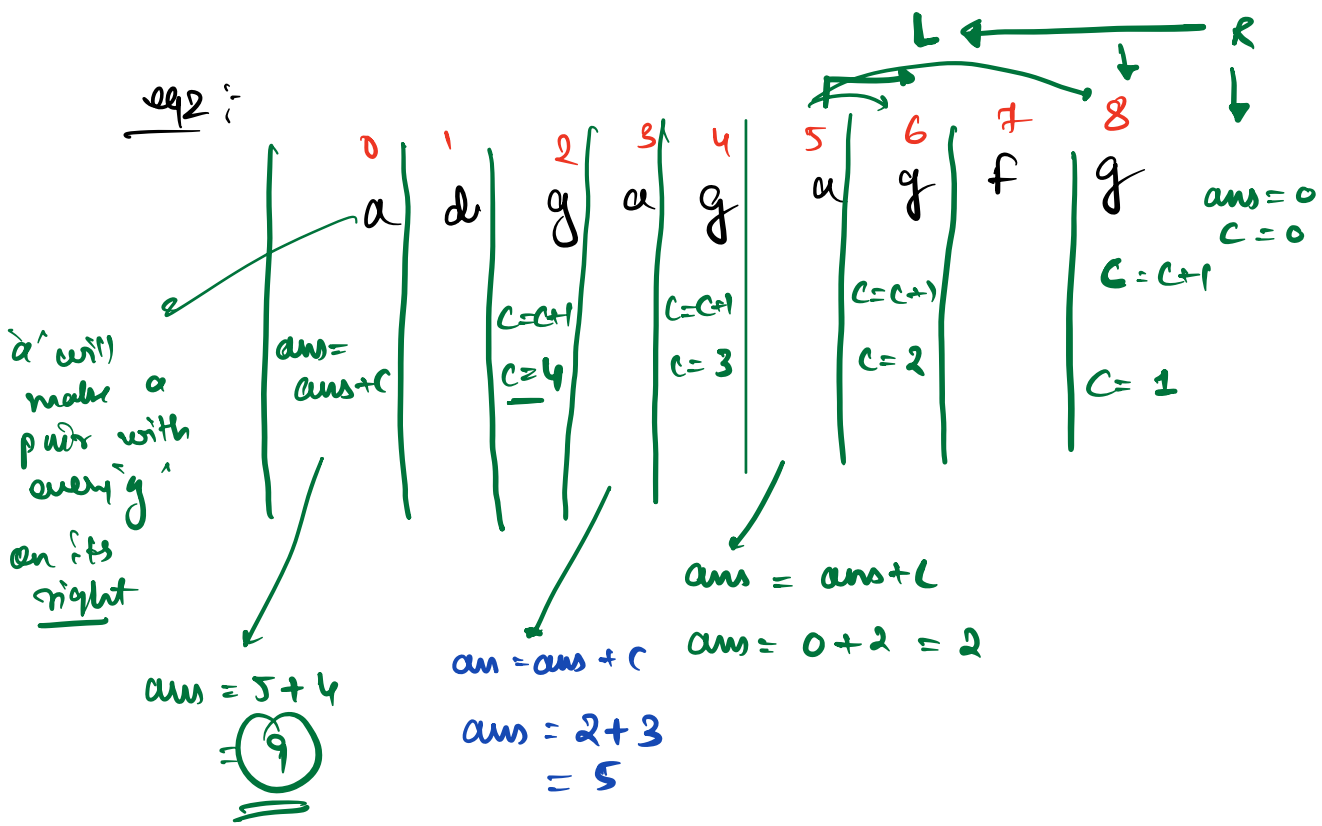
eg2 :



ans = 0

eg1 :

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



Pseudocode

```
int pair ( char ch[ ])
{   n = ch.length
    int c=0 , ans=0
    // L ← R
    for ( i= n-1 ; i>=0 ; i--)
    {
        if ( ch[i] == 'g' )
            {  c= c+1
            }
        else if ( ch[i] == 'a' )
        {
            ans = ans + c
        }
    }
    return ans
}
```

TC = O(n)
SC = O(1)

Bulbs :

Given N bulbs and their initial states, each bulb has a switch associated to it.

If we click on any switch,

→ Every bulb on right, including current bulb is flipped ($0 \rightarrow 1$, $1 \rightarrow 0$)

$1 \rightarrow$ ON

$0 \rightarrow$ off

Q: Min no. of times we need to click on switch so that all bulbs are On. (1).

Constraints

$1 <= N <= 10^5$

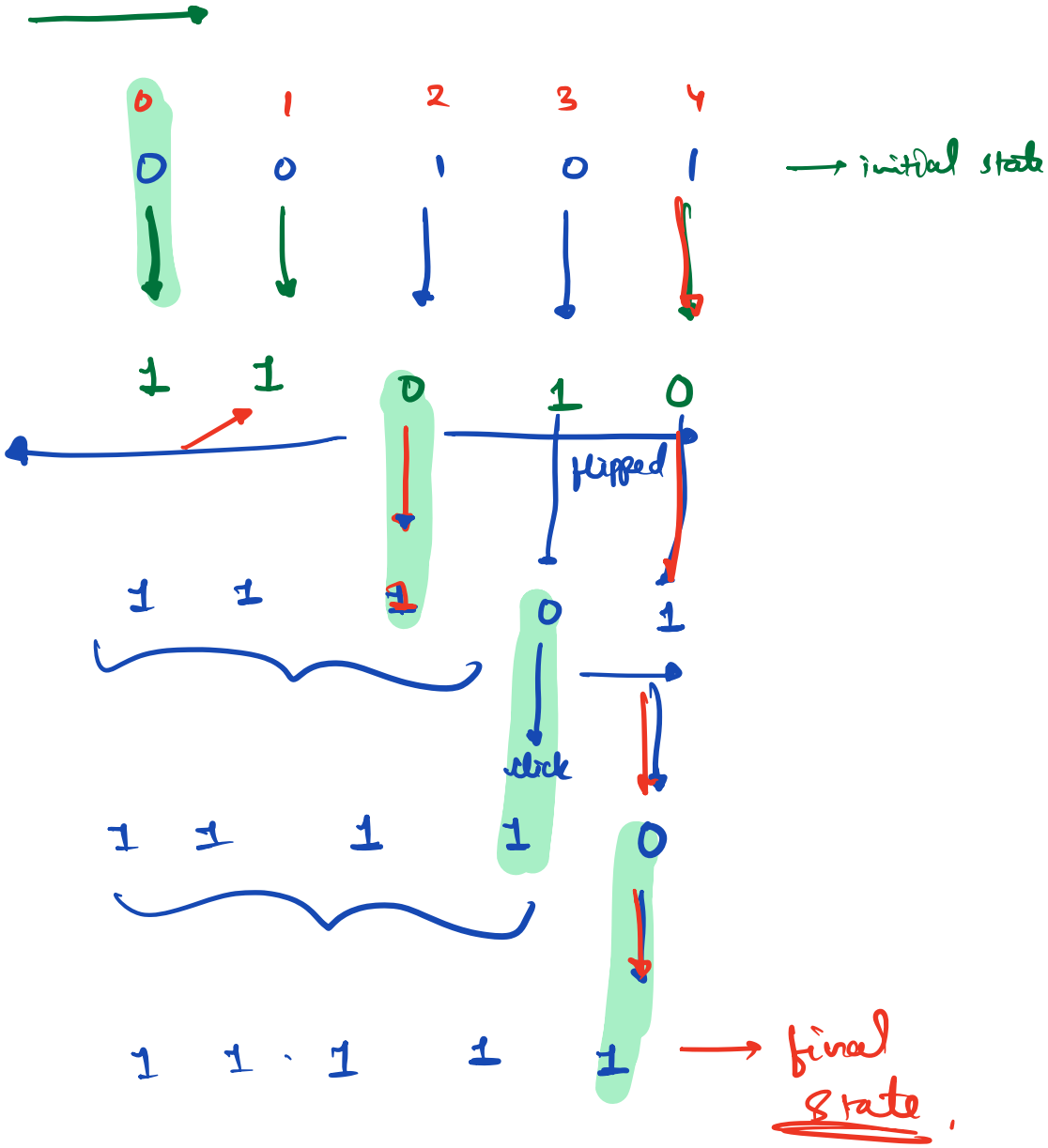
↳

1 → on

0 → off

eg 1 :-

n = 5



ans = 4

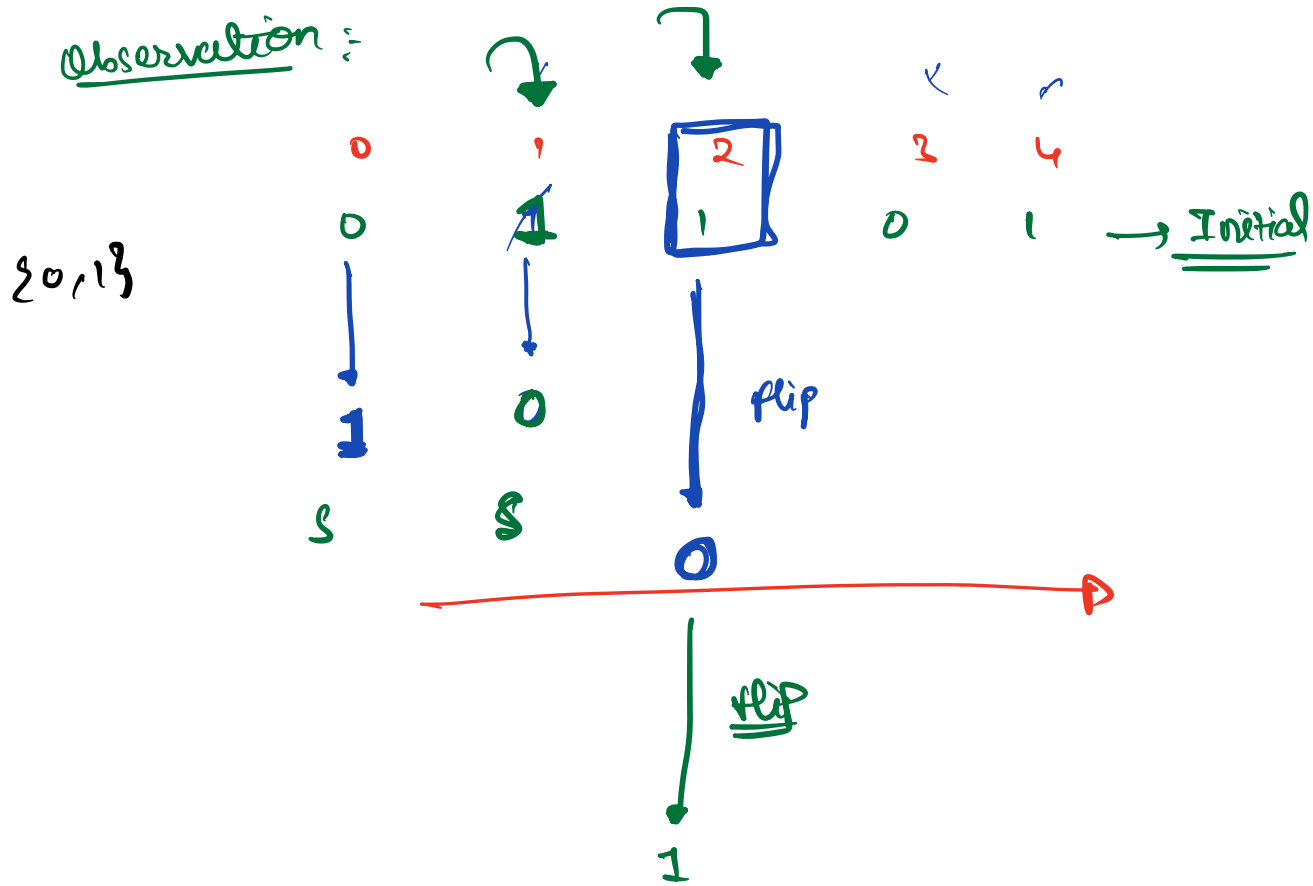
L → R

Idea :- if ith bulb is 0, flip all bulbs, click++, from i+1 to n-1.

```
int minSwitch( int b[])
{
    int c = 0
    for (i=0; i< n; i++)          // i : [0 → n-1]
    {
        // if ith bulb is off
        if( b[i] == 0)
        {
            b[i] = 1
            c = c+1
            // flip all bulbs on right of ith bulb
            for( j= i+1 ; j< n ; j++)
            {        // 1 → 0 , 0 → 1
                if (b[j]==1) { b[j]=0}
                else if(b[j]==0) {b[j]=1 }
            }
        }
    }
    return c
}
```

TC: $O(n^2)$ — TLE

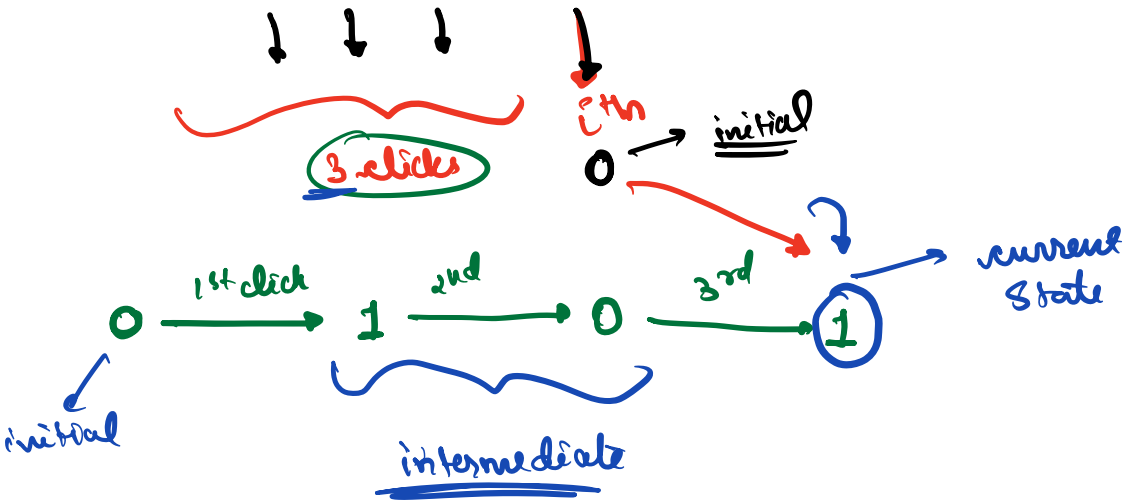
SC: $O(1)$



Obs :- whenever a click happens on left, the state of the i^{th} bulb is flipped.

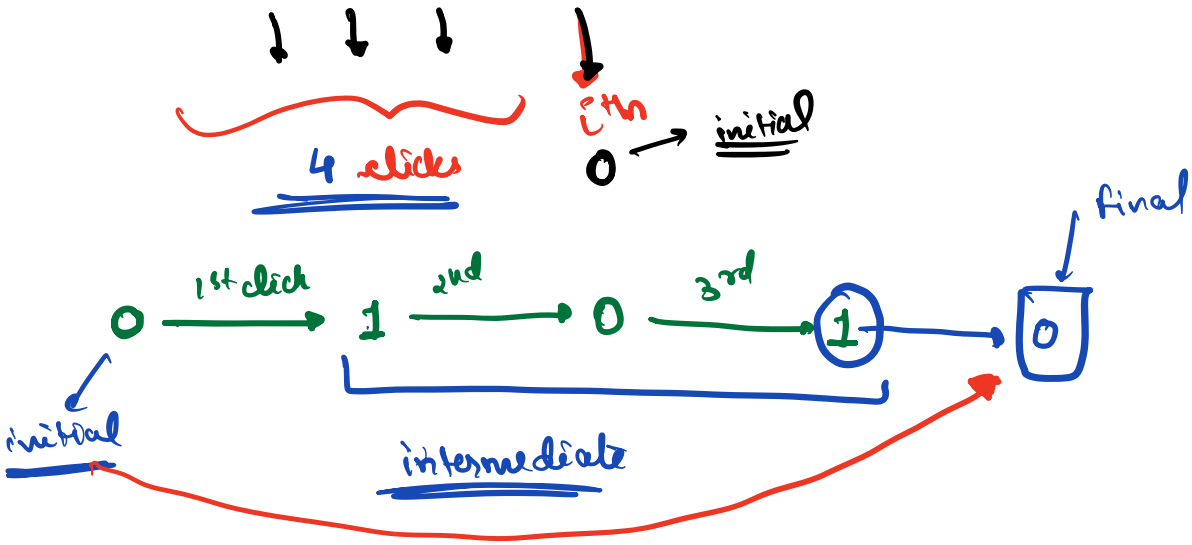
$[0, i-1]$

Case (1) If odd no. of flips on left of it



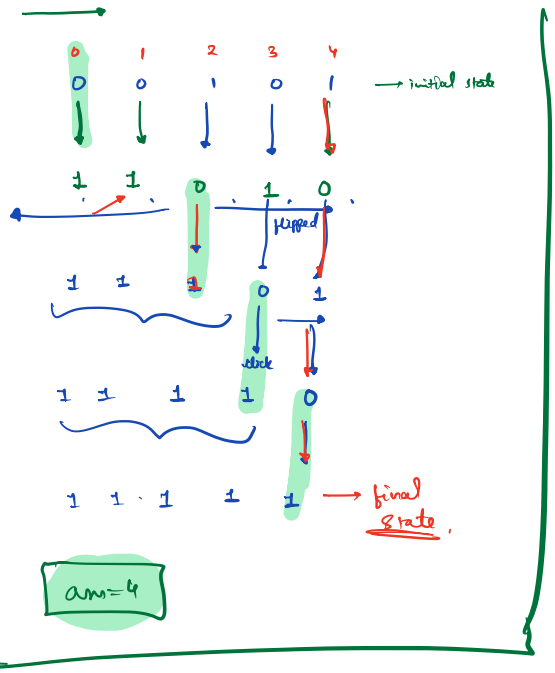
Obs :- if odd no. of clicks before reaching it → initial state is currently flipped

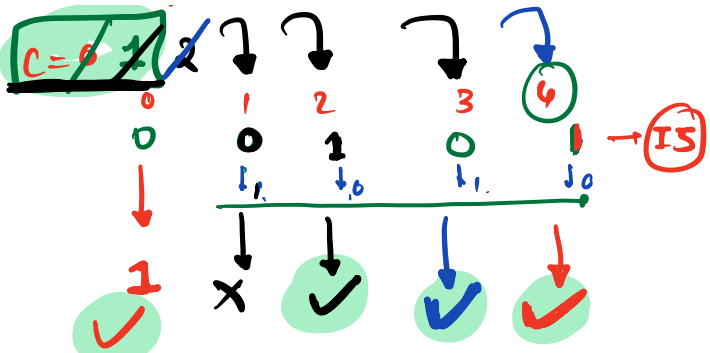
Case 2 → if even no. of flips on left of it



Obs :- if even no. of clicks before reaching it → initial state is currently same.

n=5





if c → even

curr → initial

Visualize:-

4th bulb

1 —1→ 0 —2→ 1 —3→ 0 ✓

Pseudocode :-

curr state

① If initial → 0 , even flips done → 0

clock

0 → 1 → 1 → 0 (1, 2)

1 → same → 1

② odd clicks

Initial

0 → flip → 1

curr

1 → 0 → clicked

③ Func :- Decision on ith bulb.

If curr state 1 → don't click

0 → click

Pseudocode :

```
int minSwitches (int b[])
{
    int n = b.length()
    int c = 0
    for (i=0; i<n ; i++)
    {
        // Initial state of ith bulb → b[i]
        if( b[i] == 1 and c%2 == 1)
        {
            c++
        }
        else if (b[i]==0 and c%2==0)
        {
            c++
        }
    }
    return c
}
```

iterate over every bulb and decide for it.

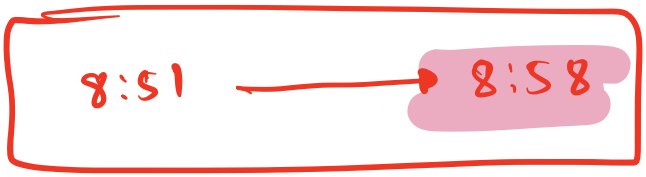
curr state is off (0)

↓

make it on by clicking

TC : O(n)

SC : O(1)


8:51
8:58

(Q.2) Leaders in a Array:

→ Given an arr[N], you have to count leaders in arr[].

Note:

① arr[i] is said leader:
if its greater than (>) max of all the elements on its left from [0, i-1].

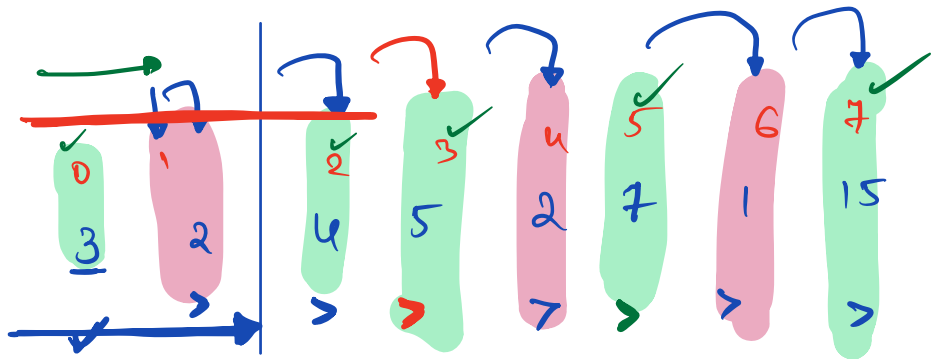
② arr[0] is considered as leader. →

Constraints:

$1 <= N <= 10^5$

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

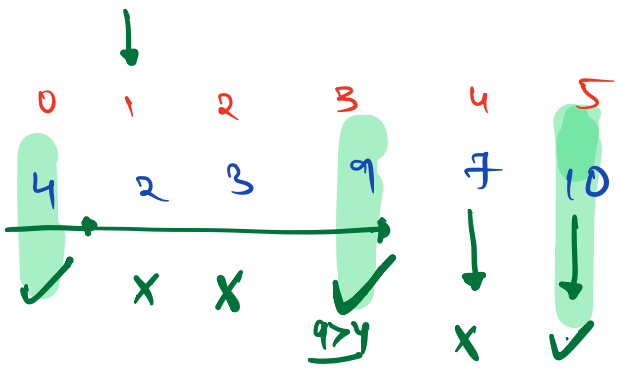
Q1:



Ans: 5

max so far on left → - 3 3 4 5 5 7 7

Q2



Ans: 3

max - 4 4 4 9 9

Approach :- ① for [i :- 0 ⟶ n-1]

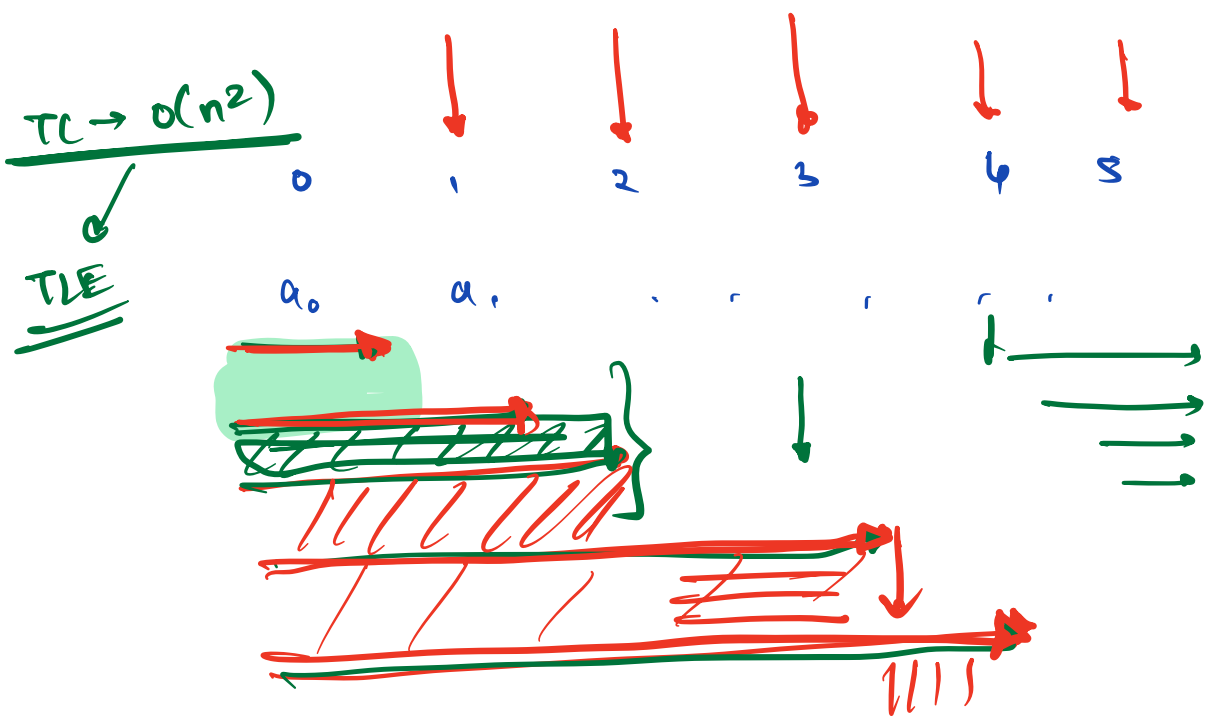
② find the max elem

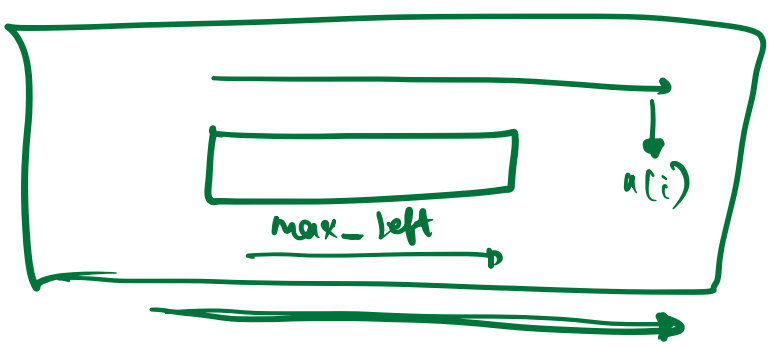
a[i] > max_left

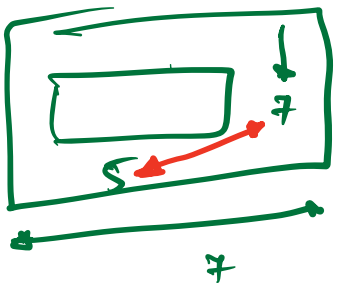
⟶ leader ++

TC → O(n²)

⟶ TLE



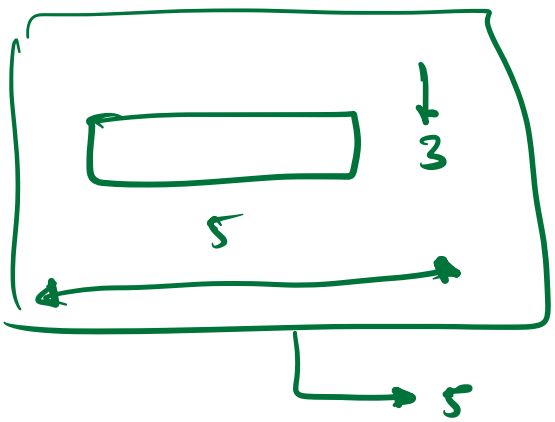




a(i) > max_left

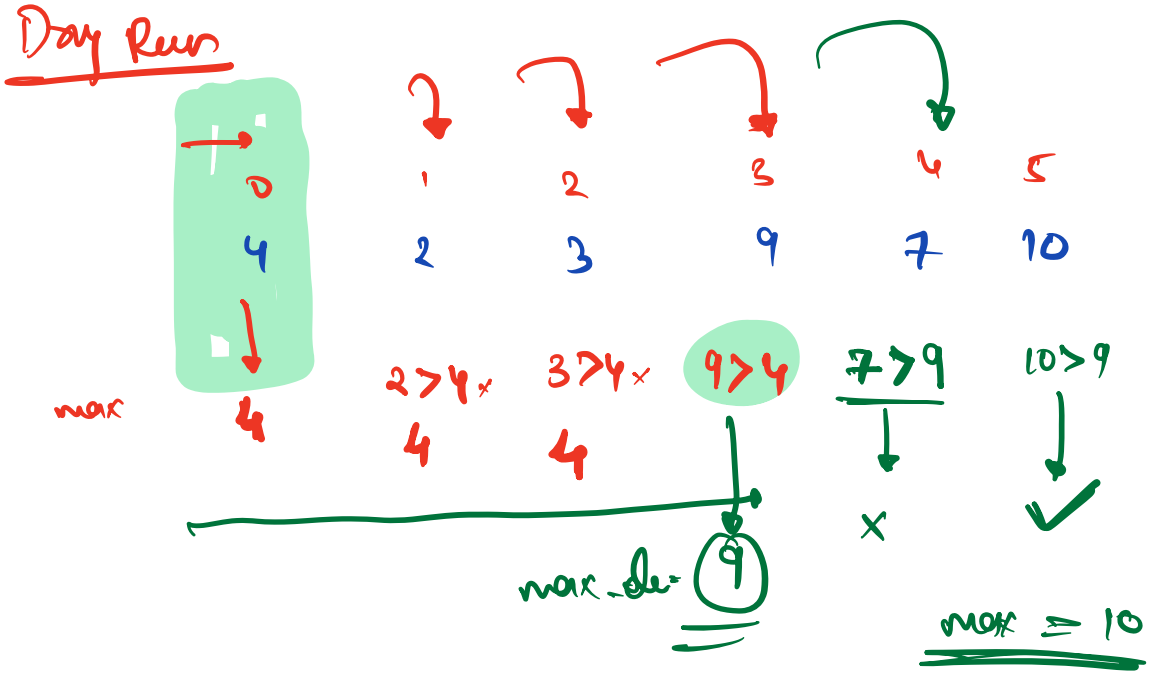
→ max_left = a(i)

else a(i) <= max_elem



→ no update

Dry Run



Pseudocode :-

```
int countleaders ( int a[] )
{
    n = a.length()
    int leaders = 1
    int max = a[0]
    for ( i=1 ; i<n ; i++ )
    {
        // check if a[i] is leader
        if (a[i] > max)
        {
            leaders++
            max = a[i]
        }
    }
    return leaders
}
```

TC : O(n)

SC : O(1)