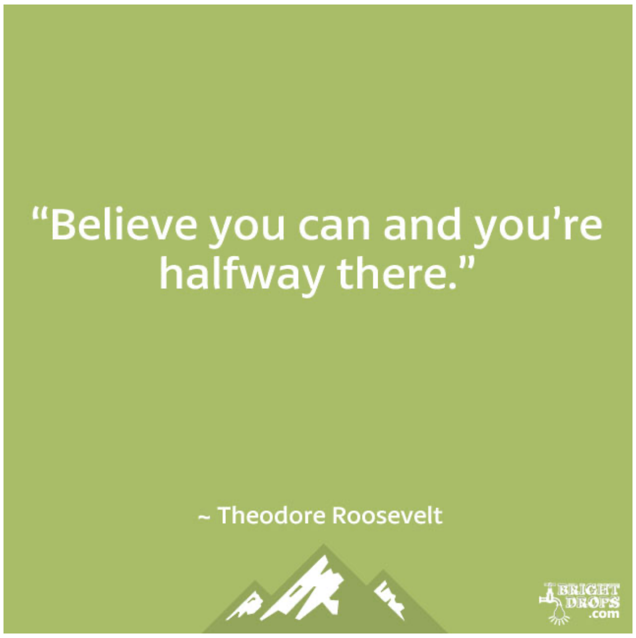
Arrays : Carry Forward



Good Evening :)

Today's content

01. Count pairs "ag"
02. Leaders in an array
03. Switching Bulbs

01. Count pairs "ag"

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

indexed

Note :- All characters are in lower case.

Constraints : $1 \leq N \leq 10^5$ N= len of array

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

Eg:- s[8] : { b a a g d c a g }

0 1 2 3 4 5 6 7

Pairs

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

(1, 7) (2, 7) ~~(7, 2)~~ X

Ans = 5

s[ ] = { b a g a g g }

0 1 2 3 4 5

Pairs

(1, 2) (3, 4)

(1, 4) (3, 5)

(1, 5)

Ans = 5

s[ ] = { 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

Idea / Brute force approach

→ Check for all pairs (i, j) & increase count when the pair is "ag"

s[ ] = { b a g a g g }
0 1 2 3 4 5

i = 0	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)
i = 1		(1, 2)	(1, 3)	(1, 4)	(1, 5)
i = 2			(2, 3)	(2, 4)	(2, 5)
i = 3				(3, 4)	(3, 5)
i = 4					(4, 5)

Pseudocode

```
int countag ( char [] s)
{
    int c=0;
    int n=s.length;

    for ( i=0 ; i<n ; i++ )
    {
        for (j=i+1 ; j<n; j++) {
            if ( s[i]=='a' && s[j]=='g' ) { c++ };
        }
    }
    return c;
}
```

TC= $O(n^2)$

SC= $O(1)$

Constraint $N = 10^5$

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

$$= (10^5)^2$$

$$= \underline{\underline{10^{10}}} \longrightarrow TLE$$

Observation → If s[i] == 'a' then only we can form an "ag" pair

Idea 2

```
int countag( char[] s)
{
    int c=0;
    int n=s.length;

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

$\boxed{\text{TLE}}$

Worst case = s[ ] = {a, a, a, a, a, a}

TC = $O(n^2)$

SC = $O(1)$

```
if (s[i]=='a')                       ------>   if (s[i]=='a')
{
    for(j=i+1; j<n; j++)
    {
        if (s[j]=='g') {c++;}
    }
}
```

} counting the no. of g in RHS

Eg:- a d g a g a g f g

count = 4

count = 3

count = 2

count = 9

Optimised Idea

Instead of going from L → R again & again

→ Come from R → L carrying the no. of 'g'

(count of pairs) ans = 0

(count of g) c = 0

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

```
int countag(char[] s)
    int n= s.length;
    int c=0, ans=0;

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

$TC = O(n)$

$SC = O(1)$

L → R

ans = 0 (count of pair)

c = 0 (count of a)

a	d	g	a	g	a	g	f	g
c=c+1 c=1		ans+=c ans=1	c=2	ans+=c ans=3	c=3	ans+=c ans=6		ans+=c ans=9

Code → {TODO}

Q2. Leaders in an Array

Given ar[N], count the no. of leaders in arr[].

ar[i] is said to be leader, if it's greater than max of all elements on left from [0 - (i-1)]

Note :- arr[0] is considered as a leader

Constraints

$1 \le N \le 10^5$

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

Eg:- ar[8] :

	3	2	4	5	2	7	-1	15
	0	1	2	3	4	5	6	7
max	3	> 3	> 3	> 4	> 5	> 5	> 7	> 7

Ans = 5

arr[6] : { 4 2 3 9 7 10 }

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

Ans = 3

Idea → For every arr[i], get max from (0 to i-1) & then compare with arr[i]

```
int countleaders (int [] ar)
    int n= ar.length;
    int leader = 1

    for (i=1 ; i<n ; i++) {
        int max = ar[0] ;
        for ( j=0 ; j<i ; j++) {
            if (arr[j] > max) { max = arr[j] };
        }
        if (arr[i] > max ) { leader ++; }
    }
    return leader
}
```

$TC = \underline{\underline{O(n^2)}}$

$SC = O(1)$

$N = 10^5 \longrightarrow (N^2) \Rightarrow (10^5)^2$

$= 10^{10} \longrightarrow$ TLE

arr = { 4 2 3 9 7 10 }

→ 4<2

→ 4<3

→ 4<9

→ 9<7

→ 9<10

Tracing

4	2	3	9	7	10
0	1	2	3	4	5
l=1 max=4	l=1 max=4	l=1 max=4	l ++ l=2 max=9	l=2 max=9	l++ l=3 max=10

```
int leaders (int [] arr , int n )
{
    int l=1
    int max=arr [0]

    for ( i=1 ; i<n; i++) {

        if (arr[i] > max) {
            l = l+1 ;
            max= arr[i] ;
        }
    }
    return l;
}
```

TC= O(n)

SC= O(1)

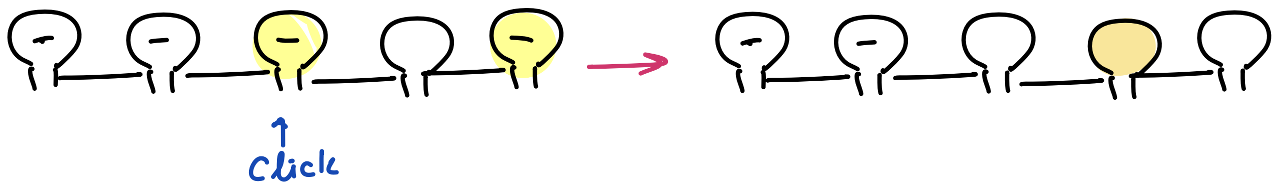
N bulbs

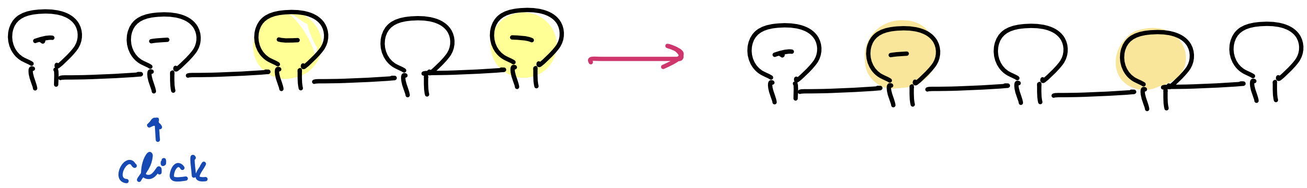
Given N bulbs & their initial state, each bulb has a switch associated to it.

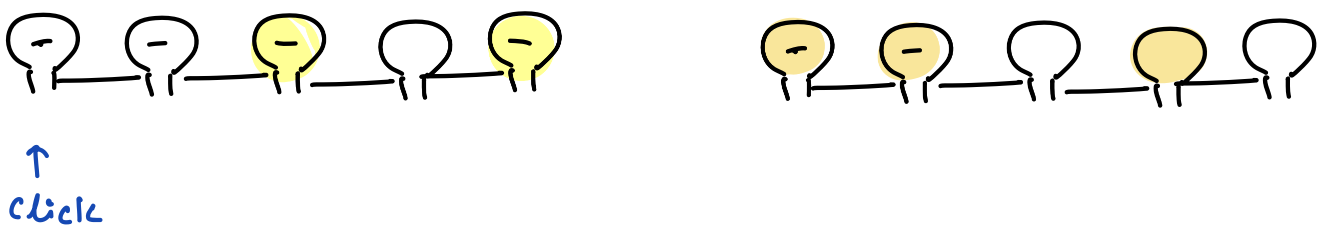
If we click on a switch, Every bulb on right including current bulb is flipped : ON $\rightleftarrows$ OFF

Constraints : $1 \leq N \leq 10^5$

Eg:-



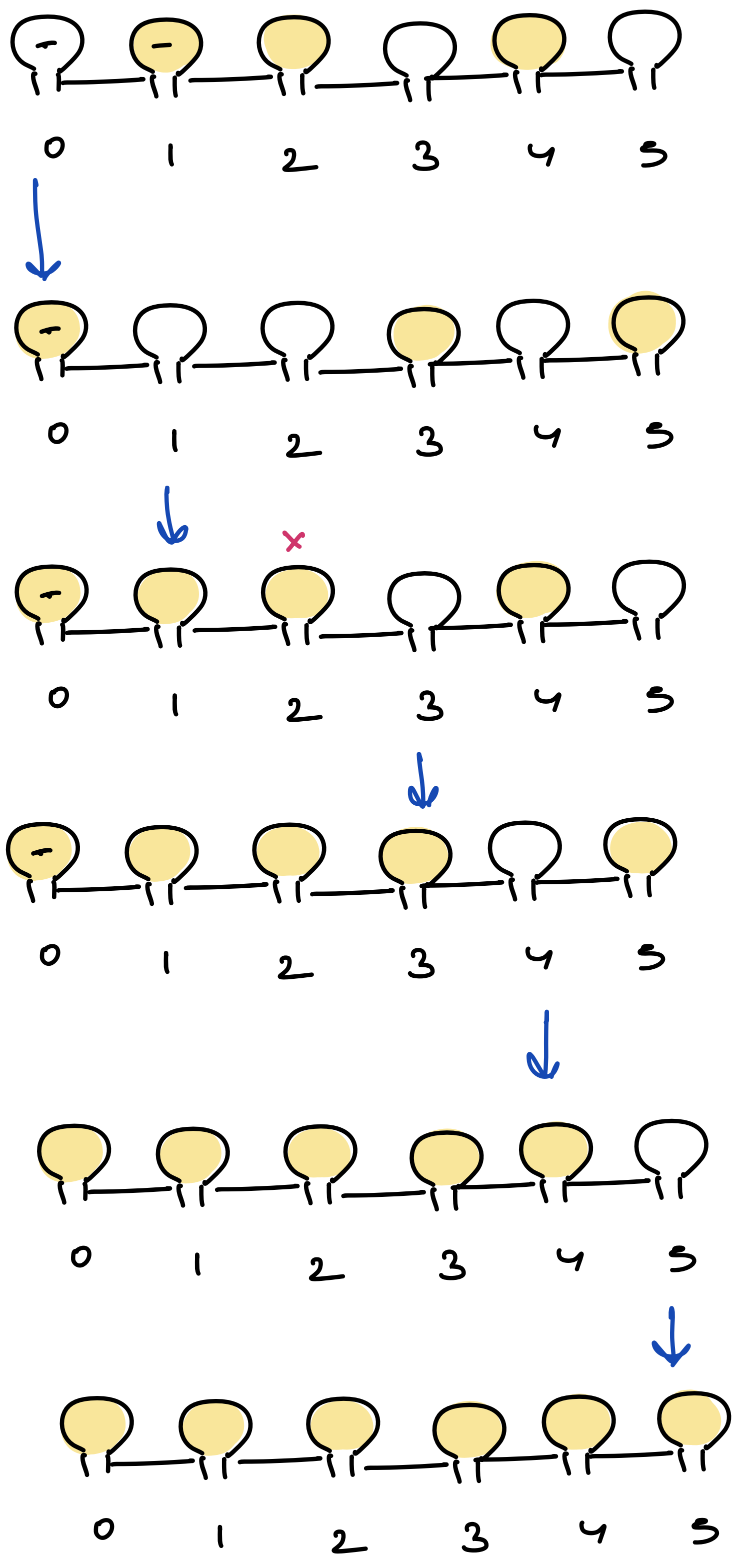




{0 0 1 0 1}

Q → Min no. of times, we need to click on switch to have all the bulbs on in our final state

Idea → Go from L→R, switch it if the bulb is off → change state of bulbs on RHS



{0 1 1 0 1 0}

↓

{1 0 0 1 0 1}

Ans = 5

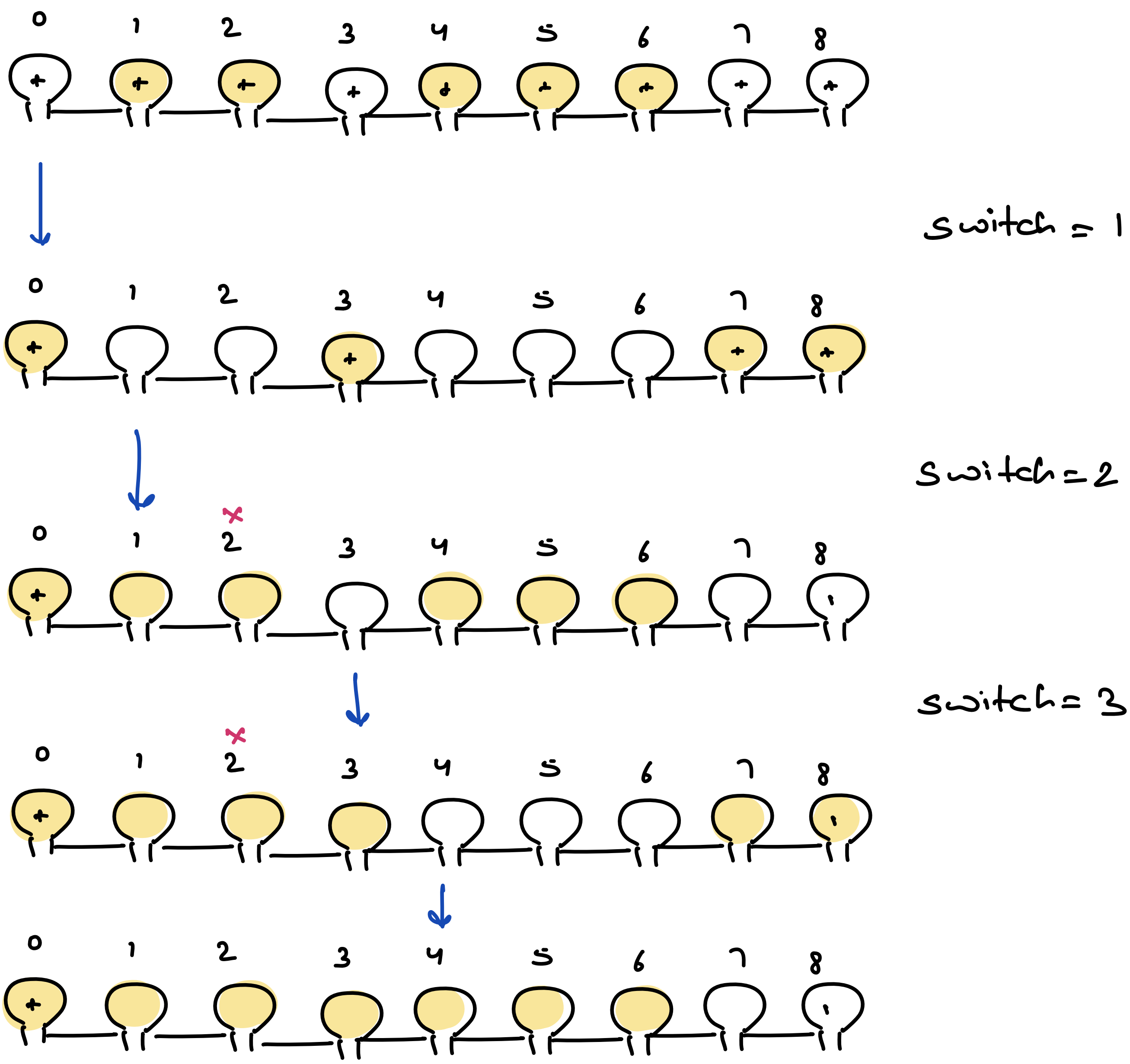
```
int minswitches ( int [] ar )
    int count = 0
    for ( i = 0 ; i < N ; i++ )
        if ( ar [i] == 0 ) {
            // switch ON the bulb
            count = count + 1;
            ar [i] = 1 ;
            for ( j = i+1 ; j < N ; j++ ) {
                if ( arr [j] == 0 ) { ar [j] = 1 }
                else if ( ar [j] == 1 ) { ar [j] = 0 }
            }
        }
    }
    return count
}
```

$TC = O(n^2)$

$SC = O(1)$

Worst case = { 0 1 0 1 0 1 }

Optimised Idea



Initial state	Flipped	Final state
OFF	3	ON

OFF → ON → OFF → ON

Initial state	Flipped	Final state
ON	4	ON

ON → OFF → ON → OFF → ON

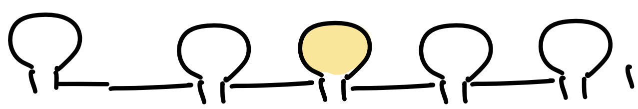
OFF 4 OFF

OFF → ON → OFF → ON → OFF

States	Flipping odd	
OFF	→	ON
ON	→	OFF

states	Flipping Even	
OFF	→	off
ON	→	ON

Tracing



Initial state	OFF	OFF	ON	OFF	off
C = 0	Even	Odd	Odd	Even	odd
Find state	OFF	ON	OFF	off	on
switch	C++ C=1	C=1	C++ C=2	C++ C=3	C=3

```
int switches (int [] arr, int n)
    int c=0

    for (i=0; i<N; i++){
        // Initial state of bulb = arr[i]

        if (arr[i]==0 && c%2==0){c=c+1}

        else if (arr[i]==1 && c%2==1){c=c+1};
    }
return c;
```

{0 1 0 1 0 1}

$c = \cancel{0}$

$\cancel{1}$

$\cancel{2}$

$\cancel{3}$

$\cancel{4}$

$\cancel{5}$

6

Doubt session

```
for ( i = 0 ; i < 2^n ; i++ )
    int j = i
    while ( j > 0 ) {
        j -= 1
    }
}
```

i	j	iterations
0	0	0
1	1	1
2	2	2
3	3	3
⋮	⋮	⋮
2^n-1	2^n-1	2^n-1

$$1 + 2 + 3 + \cdots + 2^n - 1$$

$$n = \underline{\underline{2^n - 1}}$$

$$1 + 2 + 3 + \cdots + n$$

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

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

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

$$= \frac{2^{2n} - 1}{2} \rightarrow \frac{4^n - 1}{2}$$