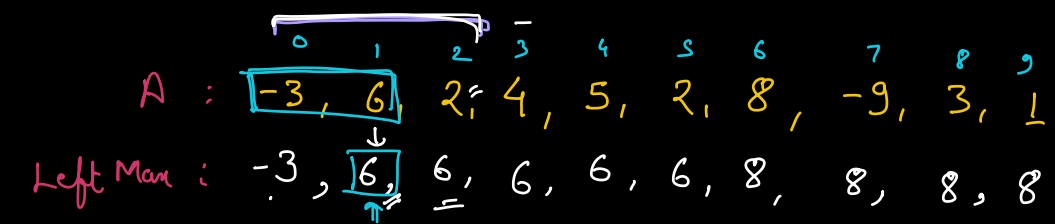
$$PS[i] = PS[i-1] + A[i]$$
$$0 \quad (i-1)$$

Q Given an array of size N.
Build an array leftMax

where leftMax[i] → The max value in the array from index 0 → i



LM[0] = A[0]

LM[1] → Max[0, 1] → Max(LM[0], A[1])

LM[2] → Max[0, 2] → Max(LM[1], A[2])

LM[3] → Max[0, 3] → Max(LM[2], A[3])

⋮

LM[i] → Max[0, i] → Max(LM[i-1], A[i])

```
LM[0] = A[0];
for(i=1; i<N; i++){
    LM[i] = max(LM[i-1], A[i]);
}
```

TC : O(N)
SC : O(N)

RightMax[i] ⇒ Max of all the elements from index i to N-1

	0	1	2	3	4	5	6	7	8	9
A :	-3	6	2	4	5	2	8	-9	3	1
RightMax :	8	8	8	8	8	8	8	3	3	1

⇒ RM[N-1] = A[N-1]

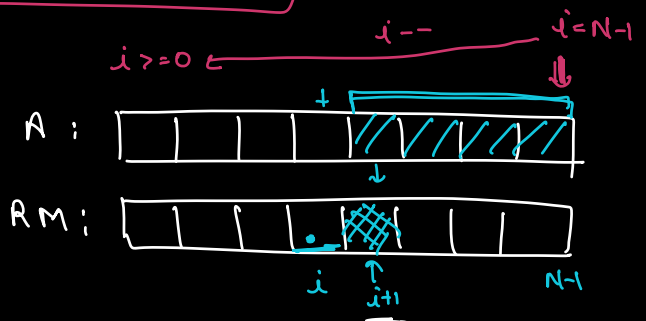
RM[N-2] = Max(N-2, N-1) → max(RM[N-1], A[N-2])

⋮

RM[2] = Max(2, N-1) → max(RM[3], A[2])

RM[1] = Max(1, N-1) → max(RM[2], A[1])

RM[i] = Max(RM[i+1], A[i])



Q Given a string of lowercase alphabets.
Return the count of pairs (i, j) such that

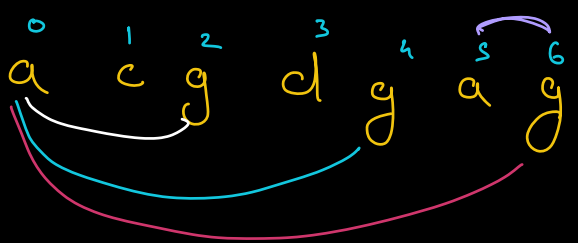
$$i < j$$

$s[i] = 'a'$
$s[j] = 'g'$] 'ag'

s : a b e g a g
0 1 2 3 4 5

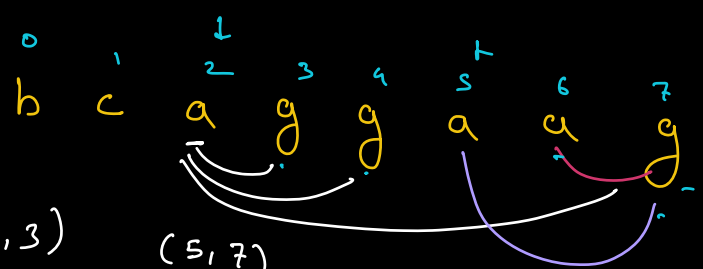
(0, 3) (4, 5)
(0, 5) ⇒ 3

Quiz



(0, 2) (5, 6) ⇒ 4
(0, 4)
(0, 6)

Quiz



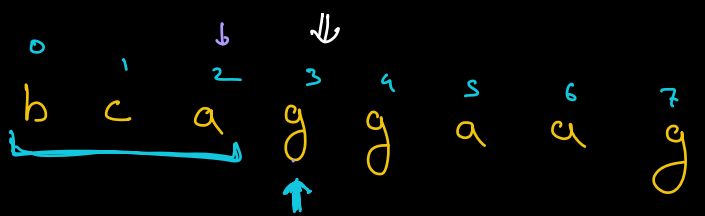
(2, 3) (5, 7)
(2, 4) (6, 7)
(2, 7)

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

TC : $O(N^2)$

(i, j)

$i < j$

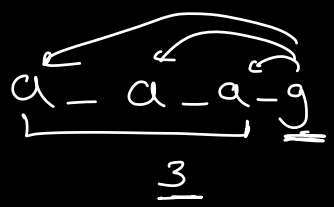


Every 'g' will make a valid pair with all the 'a' on the left of it

	a	c	b	a	g	k	a	g	g
Count(a)	1	1	1	2	2	2	3	3	3
ans	0	0	0	0	2	2	2	5	8

↑ variables (NOT array)

```
ans = 0;
Count_a = 0;
for (i = 0; i < N; i++) {
    if (S[i] == 'a') {
        Count_a++;
    }
    else if (S[i] == 'g') {
        ans = ans + Count_a;
    }
}
Ret ans;
}
```



TC : O(N)

SC : O(1) (Extra)

Amazon

Q Given an array. Return the length of smallest subarray which contains both the min & the max of the array.

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

Max (Array) → 6

Min (Array) → 1

(3, 6) ⇒ 4

1, 3, 4, 6

Quiz A: 2, 2, 6, 4, 5, 1, 5, 2, 6, 4, 1 (indices 0 to 10)

Max(A) → 6

Min(A) → 1

⇒ (8, 10) ⇒ 3 — 6, 4, 1

Quiz A: 1, 6, 4, 2, 7, 7, 5, 1, 3, 1, 1, 5 (indices 0 to 11)

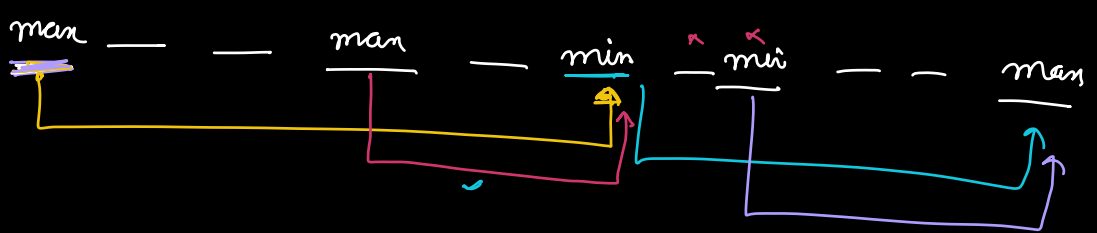
Max(A) → 7

Min(A) → 1

(5, 7) ⇒ 3 — 7, 5, 1

⇒ Ans subarray will always have max/min in the corner.

⇒ There will only be one min & one max in ans subarray.



ans = N / ∞

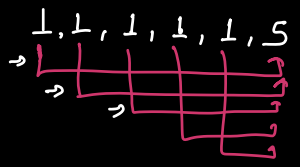
// for every min → find closest max on right

// for every max → find closest min on right

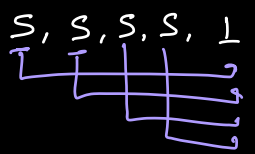
// Find Min of array → A_{min}

// find Max of array → A_{max}

```
for (i=0; i<N; i++) {
    if (A[i] == Amin) {
        for (j=i ; j<N; j++) {
            if (A[j] == Amax) {
                ans = min(ans, j-i+1);
                break;
            }
        }
    }
    else if (A[i] == Amax) {
        for (j=i ; j<N; j++) {
            if (A[j] == Amin) {
                ans = min(ans, j-i+1);
                break;
            }
        }
    }
```



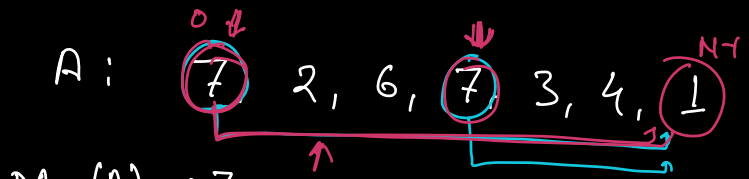
TC: $O(N^2)$



[Quiz A: 8⁰, 8¹, 8², 8³, 8⁴

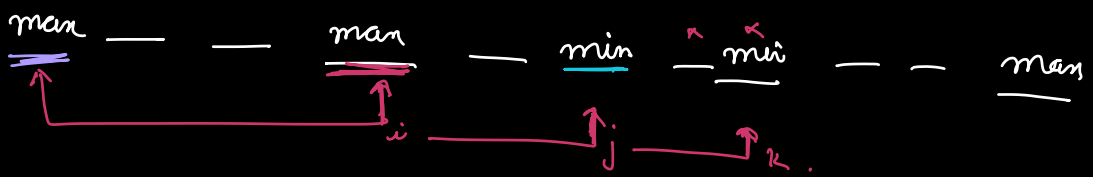
Min(A) → 8 , Max(A) → 8

[8] ⇒ 1



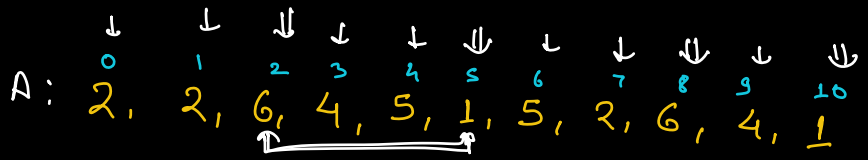
Max(A) → 7

Min(A) → 1



Last Max : ~~i~~ i

Last Min : ~~j~~ k



Last Min : ~~1~~, ~~5~~, 10

Last Max : ~~1~~, ~~2~~, 8

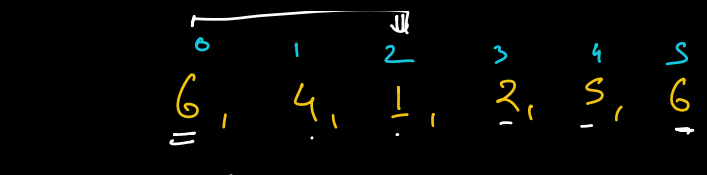
Length : ~~4~~ 3

Min → 1

Max → 6

TC : O(N)

SC (Extra) : O(1)



Last Min : ~~1~~ 0

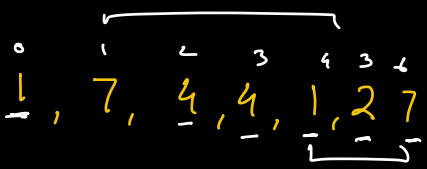
Last Max : ~~1~~ ~~2~~ 5

Length : ~~0~~ (3)

Last Min : ~~1~~ ~~0~~ ~~1~~ ~~2~~ 3

Last Max : ~~1~~ ~~0~~ ~~1~~ ~~2~~ 3

Length : ~~0~~ 1



LMin : ~~1~~ ~~0~~ 4

L Max : ~~1~~ ~~1~~ 6

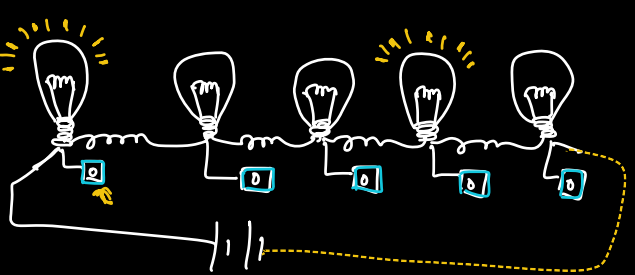
Length : ~~0~~ (2)

Q N light bulbs connected by a faulty wire.

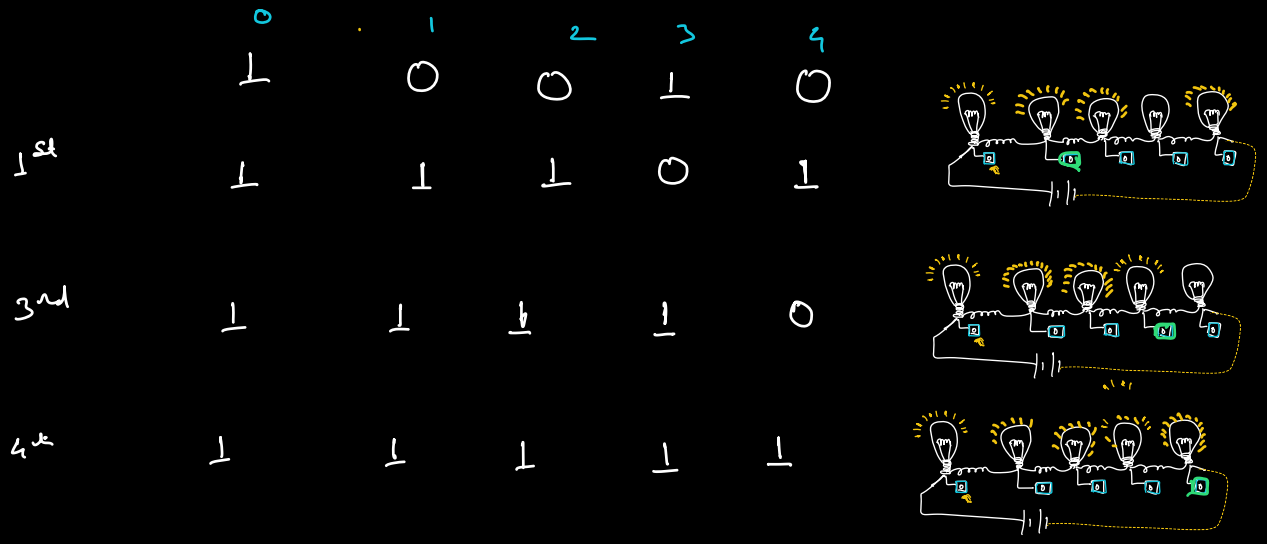
Direct-i Toggeling any i th switch → toggle all the switches on the right side.

Amazon Q1a

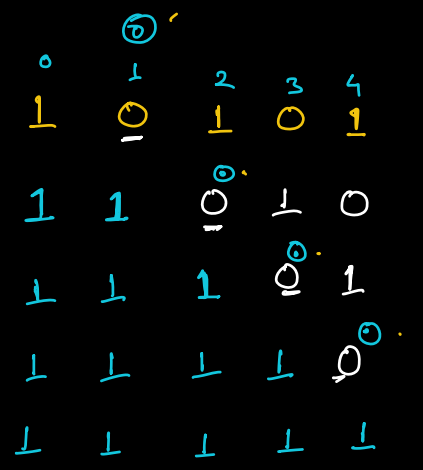
Find the min no of button press required to turn on all the bulbs.

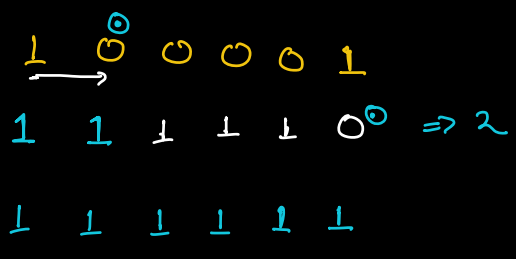


⇒ 1 0 0 1 0

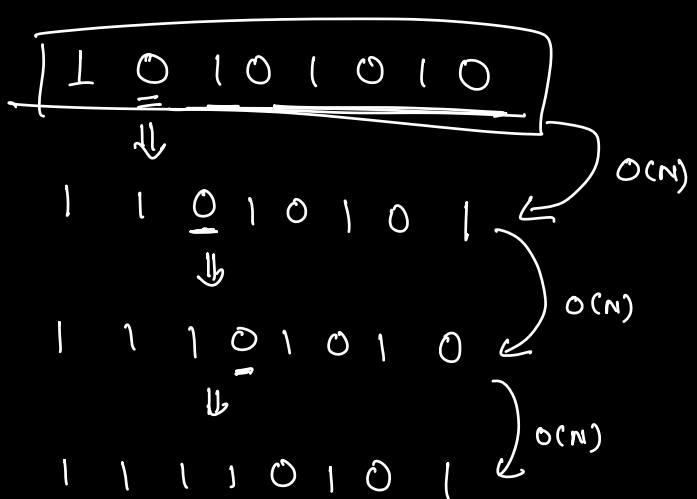


Quiz

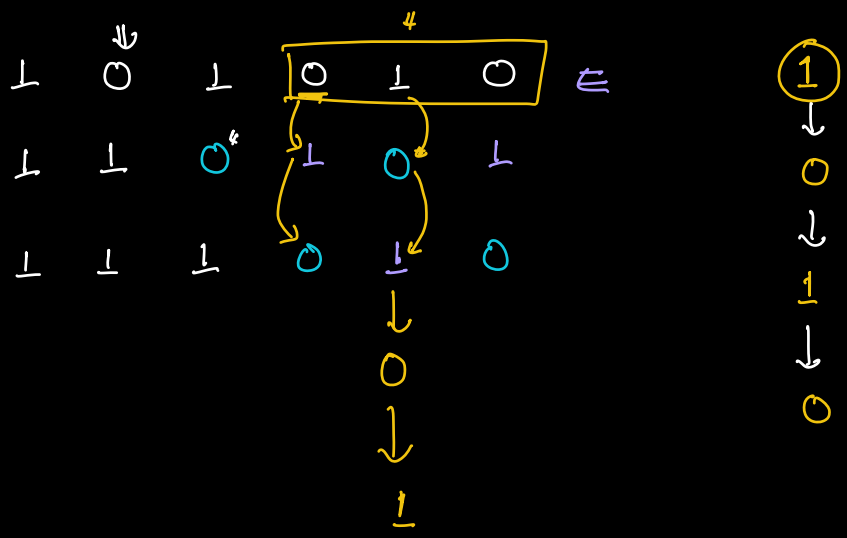




TC: $O(N^2)$



$O(N^2)$

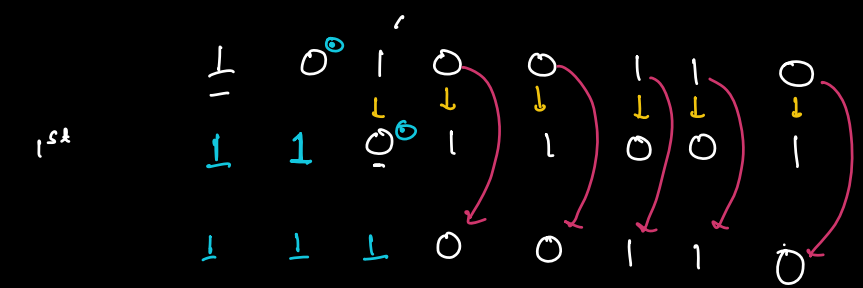


1 →(1) 0 →(2) 1 →(3) 0 →(4) 1 →(5) 0

1 →(1) 0 →(2) 1 →(3) 0 →(4) 1 →(5) 0 →(6) 1

0 →(1) 1 →(2) 0 →(3) 1 →(4) 0

0 —3171→ 1	0 —3100→ 0
1 —501→ 0	1 —420→ 1



1 0 ① ⓪ ⓪ ⓪ ①

switchPressed = 0 ; 1 2 3

```
for(i=0; i<N; i++){
    // find curState of the i^th Bulb.
    if (switchPressed % 2 == 0)
        curState = A[i];
    else // toggle the state
        // curState = A[i]==0 ? 1 : 0;
        [ if (A[i]==0)
            curState = 1;
          else
            curState = 0; ]  // 1-A[i]

    if (curState == 0)
        switchPressed++;
}
ret switchPressed;
```