Problem Solving Session

1. Pick from both sides
2. Bulbs
3. Product Array Puzzle
4. Even subarrays
5. Best time to buy & sell stock

Question 1

Given → array A, size N

You have to pick B elements. Some elements from left & rest from right.

find max possible sum of elements?

eg A = 5 -2 3 1 2 (indices 0 1 2 3 4) B = 3

Possible ways ?

[5 -2 3] 1 2 sum = 6

[5 -2] 3 1 [2] sum = 5

[5] -2 3 [1 2] sum = 8

5 -2 [3 1 2] sum = 6

let say if we have a prefix array pf[n]

if we take 0 elements from right the sum = pf[B-1]

if we take 1 element " " sum = pf[B-2] + a[n-1]

2 " " sum = pf[B-3] + a[n-1] + a[n-2]

⋮

B elements

Code

```
Create pf[n]
ans = pf[B-1]  → if we take all elements from left , sum=0
for(i=1 ; i<=B ; ++i){  → take i elements from right
    sum += a[n-i]
    if(i==B)  → when i=B , pf[B-i-1] = pf[-1]
        ans = max(ans, sum)
    else
        ans = max(ans, sum + pf[B-i-1])
}
return ans
```

Question 2 Bulbs

Given N light bulbs. Each bulb has a switch

If you flip a switch of a bulb, it changes the state of all bulbs in the right.

Given initial state of bulb, find minimum number of switches to turn on all bulbs.

1: ON 0: off

eg A = 0 1 0 1

↓flip

1 0 1 0

↓flip

total flips = 4

1 0 1 1

↓flip

1 1 0 0

↓flip

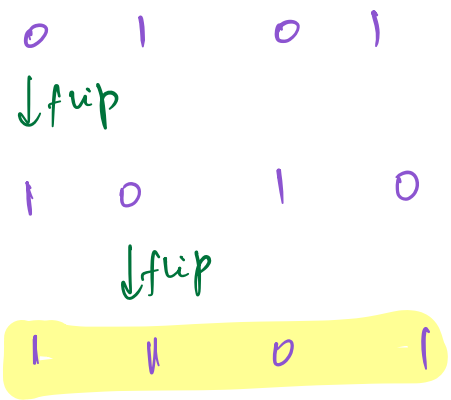
1 1 1 1

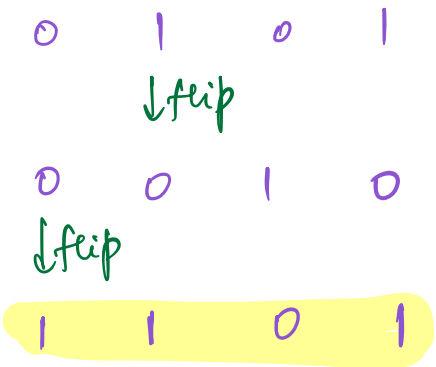
Observation 1: You will never need to press the same switch twice?

So max press of switches = N

Observation 2: Order of pressing switch does not affect final state.

eg





We can go from left → right, press switch if bulb is off.

code

```
ans=0 → # of times switches are pressed
for (i=0; i<n; ++i) {
    if (ans%2==0) { → all bulbs are in initial state from [i,n-1]
        if (a[i]==0) {
            ++ans → curr. bulb is off
        }
    }
    else { → all bulbs are in opp. state from [i,n-1]
        if (a[i]==1) {
            ++ans
        }
    }
}
```

Question 3 Product Array Puzzle

Given array A, find the value of (product of all elements) / A[i] for all i.
without using division.

Eg A = [1 2 3 4 5] product of all = 1×2×3×4×5 = 120

output = [120 60 40 30 20]

Code

```
for (i=0; i<n; ++i) {
    product=1
    for (j=0; j<n; ++j) {
        if (j != i)
            product *= a[i]
    }
    print (product)
}
```

✓ Correct but inefficient

TC : $O(N^2)$

Optimization

let say pf[i] = product of all elements till i

= a[0] × a[1] × ... × a[i]

sf[i] = a[i] × a[i+1] × ... × a[n-1]

$$ans[i] = pf[i-1] * sf[i+1]$$

Code

```
pf[n] , sf[n] , ans[n]

pf[0]=a[0],  sf[n-1] = a[n-1]

for (i=1; i<n; ++i ) {
    pf[i] = pf[i-1] * a[i]
}
for (i=n-2; i>=0; --i ){
    sf[i]= sf[i+1] * a[i]
}
for (i=0; i<n; ++i) {
    if(i==0)
        ans[i] = sf[i+1]
    else if( i==n-1)
        ans[i] = pf[i-1]
    else
        ans[i]= pf[i-1] * sf[i+1]
}
```

TC : O(N)

SC : O(N)

for i=0 out of bound

pf[i-1] = pf[-1]

for i=n-1

sf[i+1] = sf[n]

Question 4 Even Subarrays

Given array A,

Check if we can divide array in 1 or more subarray of **even length** such that **first & last** element of each subarray is **even**.

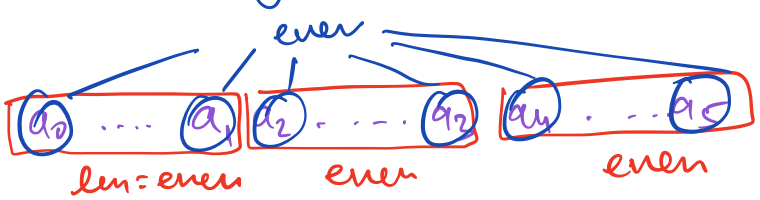
Return — YES/NO

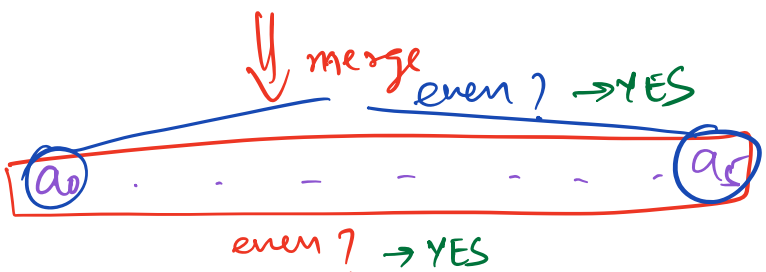
eg A = [2 4] [2 6]

[2 4 8 6]

Observation 1: If we can divide array into 1 or more subarray, we can do the same with only 1 sub array.

eg





So, If **array length is even** and **first & last element is even** then **ans = YES**.

Code

```
def EvenSubarray(a):
    n = a.length
    if(n%2==0 && a[0]%2==0 && a[n-1]%2==0)
        return "YES"
    return "NO"
```

Question 5 Best Time to Buy & Sell Stocks I

array A, A[i] → price of stock at i^{th} day.

You can do at most one transaction.

find the max profit possible ? Min profit = 0

Eg A = 1 4 5 2 4

1 ↑ Buy, 5 ↑ sell

profit = 5 − 1 = 4

Observation : Buy should be before Sell

if I buy at index i, then I can sell at any point from [i+1, n-1]

⇓

for max profit = max(a[i+1] ... a[n-1])

Sell at i=1 ⇒ Buy at i=0

Sell at i=2 ⇒ Buy at i=0,1 ⇒ Buy at min(a[0],a[1])

Sell at i=3 ⇒ Buy at min(a[0]...a[2])

⋮

Sell at i=n-1 ⇒ Buy at min(a[0]...a[n-2])

Prefix Min

Create pf[n] st.

pf[i] = min(a[0] ... a[i])

Code

```
Create pf[n]
ans=0
for(i=1; i<n; ++i) {        → sell at i
    profit = a[i] - pf[i-1];   → buy at min of [0...i-1]
    ans = max(ans, profit);
}
return ans
```

TC: O(N)

SC: O(N)

Use carry forward

```
curr_min = a[0] , ans=0

for (i=1; i<n; ++i) {
    profit = a[i] - curr_min
    ans = max(ans, profit)
    curr_min = min(curr_min, a[i])
}
return ans
```

TC: O(N)
SC: O(1)