Questions

- → Pick from both sides
- → Bulbs
- → Even Sub-arrays
- → Alternating sub-arrays
- → Good sub-arrays. (idea)

Q1) Given an array of size N. You have to pick B elements. You can select some elements from left end and some elements from right end to get the maximum sum.

N = 10.

Eg → arr = { 4 3 -2 5 6 -9 1 8 -1 2 }

B = 4

arr	4	3	-2	5	6	-9	1	8	-1	2
index	0	1	2	3	4	5	6	7	8	9
pSum	4	7	5	10	16	7	8	16	15	17
index	0	1	2	3	4	5	6	7	8	9

$\boxed{ans = 13}$

17 - pSum[10-4-1]

17 - pSum[5] = 17-7 = 10

	left	right	
0	0	4	total - pSum[n-4-1]
pSum[1-1]	1	3	total - pSum[n-3-1]
pSum[2-1]	2	2	total - pSum[n-2-1]
pSum[3-1]	3	1	total - pSum[n-1-1]
pSum[4-1]	4	0	total - pSum[n-0-1]

⇓

⇒ pSum[left-1]

pSum[n-1] - pSum[n-right-1]

pseudo-code. ① Create pSum[]

```
left = 0 , right = B , ans = 0

while( left <= B) {
    if ( left >= 1 )
    ans = Max( ans, pSum[left-1] + pSum[n-1]
                     - pSum [n-right-1]) ;
    } else {
    ans = max (ans, pSum[n-1] - pSum[n-right-1]);
    }

    left += 1
    right -= 1
}
```

$\boxed{N \geq B}$

T.C → O(N+B) → O(N)

S.C → O(N) → O(1) modify the given array.

{Todo → Try to solve by just taking 2 variables → lsum and rsum. }.

② Bulbs.

arr →

0	1	0	0	1	1	0	1

0 → bulb is off
1 → bulb is ON.

All bulbs are connected through a circuit & the circuit is faulty → All the bulbs on r.h.s will be toggled.

Min no. of switch to be pressed such that all the bulbs are finally "ON".

arr → { 1 0 1 0 1 } [ans = 4]

{ 1 1 0 1 0 }

{ 1 1 1 0 1 }

{ 1 1 1 1 0 }.

{ 1 1 1 1 1 }

Brute force.

```
count = 0
for( i → 0 to n-1) {
    if ( arr[i] == 0) {
        count += 1 ;  arr[i] = 1
        for( j → i+1 to N-1) {
            arr [j] = 1 - arr[j]
        }
    }
}
return count ;
```

$T.C \to O(N^2)$
$S.C \to O(1)$

Optimisation → {observation}

Eg : { 1 0 0 1 1 0 1 0 }

{ 1 1 1 0 0 1 0 1 }   count = 1

{ 1 1 1 1 1 0 1 0 }   count = 2

{ 1 1 1 1 1 1 0 1 }   count = 3

{ 1 1 1 1 1 1 1 0 }   count = 4

Conclusion : state of bulb is only dependent on count.

if count is even → state of bulb will be same as that is present in original array.

if count is odd → state will be toggled.

pseudo-code.

```
count = 0 ;

for( i → 0 to N-1) {
    state = 0
    if ( count % 2 == 0) state = arr[i]
      else              state = 1 - arr[i]
    if ( state == 0) {
      count += 1 ;
    }
}
return count
```

$\boxed{\begin{array}{l} T.C \to O(N) \\ S.C \to O(1) \end{array}}$

Q3) **Even Sub-arrays**

Given an arr[ ]. Find whether it is possible to divide the array into one or more subarrays of even length such that first and last element of all sub-arrays will be even.

Eg1 { 4 6 2 8 10 } : NO

indices: 0 1 2 3 4

length % 2 == 1,
⇓
"NO"

Eg2 { 2 4 6 8 10 12 14 } : NO

indices: 0 1 2 3 4 5 6

Eg3 { 3 4 6 8 } : "NO"

indices: 0 1 2 3

Eg4 { 4 16 28 9 } : "NO"

indices: 0 1 2 3

Eg5 { 4 3 5 7 9 11 17 20 }

indices: 0 1 2 3 4 5 6 7

if (N % 2 == 0 and arr[0] % 2 == 0 and
arr[N-1] % 2 == 0)
⇒ "YES"

else
⇒ "NO"

arr → [4, 6, 4, 2, 5, 7, 2, 1]
0 1 2 3 4 5 6 7

[4, 6, 4, 2, 5, 7, 2, 2]

[4, 6, 5, 2, 5, 7, 2, 2]

include all the elements of array in your sub-arrays.

Q4.) **Alternating Sub-arrays**

Given an array containing 0's and 1's & an integer B.
Find all the indices of array A that can act as a center of $(2*B+1)$ length 0-1 alternating subarray.

$\{0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\}$
$\{1\ 0\ 1\ 0\ 1\ 0 - -\}$

Eg: $\{1\ 0\ 1\ 0\ 1\}$, $\boxed{B=1}$
indices: 0 1 2 3 4

$(2*B)+1 = 3.$

ans→ [1, 2, 3]

$\boxed{B=2}$ $\boxed{len \to 5}$

Eg: $\{1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\}$
indices: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

3 sub-arrays which are 0-1 alternating.

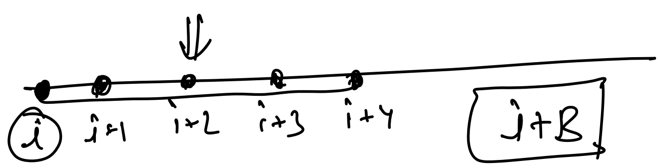
ans→ {4, 12, 13}

// consider all the sub-array of length $(2*B)+1$.

Total no. of sub arrays = $\frac{N(N+1)}{2} \approx N^2$

$\boxed{B=2}$

len = 5



$\boxed{i+B}$

pseudo-code.

```
void AlternatingSubarrays ( arr, N , B ){
        len = (2*B) + 1
        for ( i = 0 ; i < N-len ; i++) {
            prev = -1 , flag = true
            for( j = i ; j < i+len ; j++){
                if (arr[j] == prev){
                    flag = false
                    break
                }
                prev = arr[j]
            }
            if (flag == true)   print ( i+B );
        }
}
```

prev = ~~-1~~ ~~1~~ 0 1

flag = true

[1 0 1 0 1] , B=1

0 1 2 3 4

len=3.

$T.C \rightarrow O(N^2)$

$S.C \rightarrow O(1)$

Q. **Good Sub.arrays** Given an arr[N] and B.

- → length → even, sum of all elements < B
- → length → odd, sum of all elements > B

Count of good sub arrays in array?

Eg: { 6 4 3 7 8 1 } $\boxed{B=15}$

indices: 0 1 2 3 4 5

```
int goodSubarrays ( arr, N, B) {
    count = 0;
    for( i = 0 ; i < N ; i++){
        sum = 0
        for( j = i ; j < N ; j++){
            sum += arr[j] ;
            len = j - i + 1 ;
            if ( len % 2 == 0 && sum < B) count += 1
            if ( len % 2 == 1 && sum > B) count += 1
        }
    }
    return count ;
}
```

$\boxed{\begin{array}{l} T.C \rightarrow O(N^2) \\ S.C \rightarrow O(1) \end{array}}$

Product Array

arr →

10	20	3	50	4
0	1	2	3	4

[print product of all the element except itself.]

prefix →

10	200	600	30000	120000
0	1	2	3	4

suffix →

120000	12000	600	200	4

arr →

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

$\boxed{B = 4}$

$N - B = 9 - 4 = 5$

left	right
0	4
1	3
2	2
3	1
4	0

lsum	rsum
0	18
3	12
8	3
6	7
14	0

+5, -2, +8

-9, -(-4), -7

lsum + rsum

[0 + 18]

[3 + 12]

18. ⇐ [8 + 3]

[6 + 7]

[14 + 0]

pseudocode.

```
lsum = 0 ; rsum = 0
for ( i = N-B ; i < N ; i++) {
        rsum += arr[i]
}
ans = lsum + rsum ;        [0 from left , B from right]
for ( int i = 0 ; i < B ; i++) {
        lsum = lsum + arr[i]
        rsum = rsum - arr[N-B-i]
        ans = max( ans , lsum + rsum );
}
return ans;
```

```
for (i = 3 ; i < n/3 ; i += 3) {
    for (j = 2 ; j < n/2 ; j += 2) {
        ____
        ___
    }
}
```

i	j	iterations
3	(2, n/2)	$n/4$
6		$+ \; n/4$
9		$+ \; n/4$
|		$+ \; \vert$
$n/3$		$+ \; n/4$

x

$2 \quad 4 \quad 6 \quad 8 \quad - - - \quad n/2$

K

N^{th} term $\Rightarrow a + (n-1)d$

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

$$K = \frac{n-4}{4} + 1$$

$$K = \frac{n - \cancel{4} + \cancel{4}}{4}$$

$$\boxed{K = \frac{n}{4}}$$

$$\boxed{x \cdot \frac{n}{4}}$$

total no. of iterations.

$$\Rightarrow \frac{n}{9} \cdot \frac{n}{4}.$$

$$\Rightarrow \frac{n^2}{36}.$$

$$\boxed{T.C \rightarrow O(n^2)}$$

$$\underbrace{3,\ 6,\ 9,\ 12,\ -\ --\ n/3}_{x}.$$

$$(x-1)\,3 = \frac{n}{3} - 3$$

$$(x-1) = \frac{n-9}{9}$$

$$x = \frac{n-9}{9} + 1 = \frac{n\cancel{-9}+\cancel{9}}{9} = \frac{n}{9}.$$

arr → { 3, 4, 9, 6, 7, 2 }.

```
max = arr[0]
for( int i = 1 ; i < N ; i++){
        if (arr[i] > max) max = arr[i]
}

smax = 0
for ( int i = 0 ; i < N ; i++) {
        if (arr[i] != max && arr[i] > smax){
            smax = arr[i]
        }
}
```