content for today

- Pick from Both sides
- ✓ Bulbs
- Alternating Sub-arrays.
- ✓ Even Sub-arrays
- ✓ Good sub-arrays (# discuss)
- ✓ Sub-arrays with least Average of size K (# discuss)

Q: Given an array of size N.

You have to pick B elements in total. (some from left and some from right) to get the maximum sum.

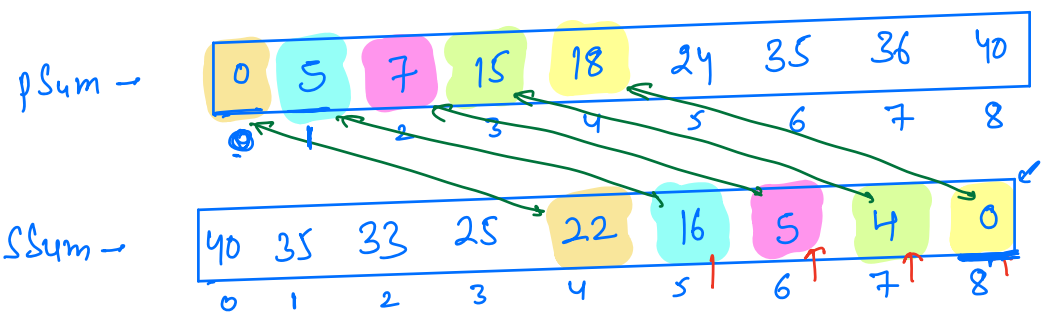
→ possibly it can zero.

a: [5 2 8 3 6 11 1 4] , B = 4

(indices: 0 1 2 3 4 5 6 7)

left	right	
4	0	⇒ 5+2+8+3 = 18
3	1	⇒ 5+2+8+4 = 19
2	2	⇒ 5+2+1+4 = 12
1	3	⇒ 5+11+1+4 = 21
0	4	⇒ 6+11+1+4 = 22

ans = 22.



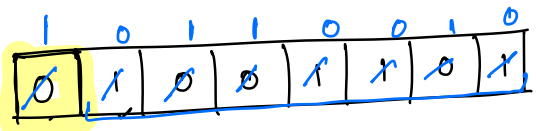
```
psum = new int[n+1], ssum = new int[n+1];
for( i=1; i<=n; i++){
    psum[i] = psum[i-1] + a[i-1];
}
for( i = n-1; i>=0; i--){
    ssum[i] = ssum[i+1] + a[i];
}

ans = INT_MIN;
for( int i=0; i<=B; i++){
    ans = max( ans, psum[i] + ssum[n-B+i]);
}
```

T.C → O(N)
S.C → O(N)

{ # to do → O(1) space complexity. }

① Bulbs.

arr → 

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 pressing such that all the bulbs are finally "ON".

arr → 1 0 1 0 1

1 1 0 1 0 count → 4

1 1 1 0 1

1 1 1 1 0

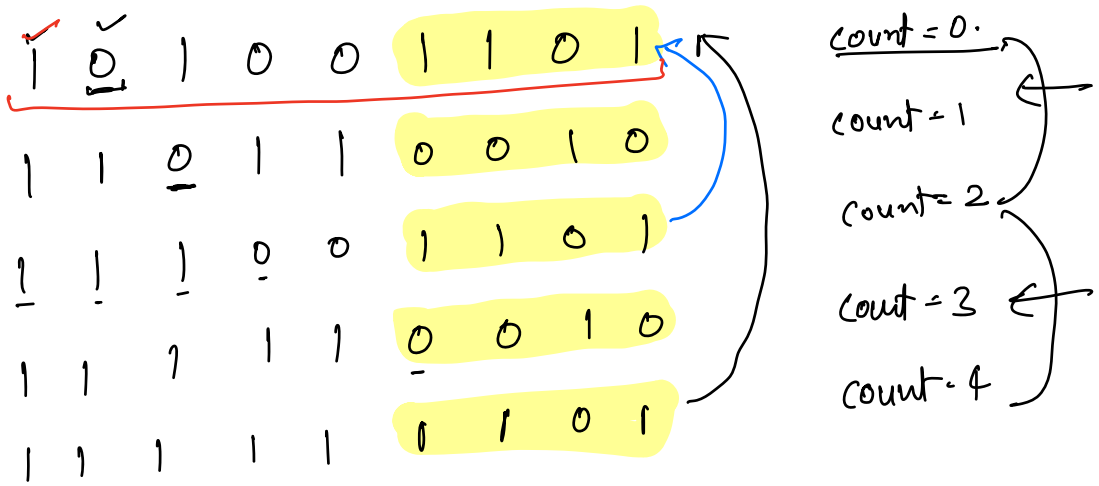
1 1 1 1 1

pseudo-code

```
count = 0 ;
for ( i = 0 ; i < n ; i++ ) {
    if (a[i] == 0) {
        count++ ;
        for ( j = i ; j < n ; j++) {
            a[j] = 1 - a[j] ;
        }
    }
}
return count ;
```

T.C → O(N²)
S.C → O(1)

optimization.



- state of a bulb is dependent on count.
- if count is even ⇒ state will be similar to what it was in original array.
  odd ⇒ state will be toggled.

pseudo-code.

```
count = 0, curr = 0;
for ( i=0 ; i < N ; i++){
    if (count %2 == 0){
        curr = a[i];
    }
    else {
        curr = 1 - a[i];
    }
    if (curr == 0){
        count++ ;   //switching ON the current bulb.
    }
}
return count;
```

T.C → O(n)
S.C → O(1)

Q: Given an array of size N which contains only 0's & 1's. Find indices of all the alternating sub-arrays with length = 2*B+1. [B is given]

alternating sub-arrays

[0 -1 -0 -1]
[1 -0 -1 -0]

Eg: n=5 [1 0 1 0 1], B = 1, length = (2*1)+1 = 3.

0 1 2 3 4

→ Consider all sub-arrays with length = (2*B)+1.

```
len = (2*B)+1;   list<int> ans;
for( i = 0 ; i < n-l+1 ; i++ ){
    prev = -1 , flag = true ;
    for( j = i ; j < i + len ; j++){
        if (a[j] == prev) {
            flag = false;
            break;
        }
        prev = a[j];
    }
    if (flag == true) ans.add(i+B);
}
```

n:

1 0 1 0 1 0 1

prev = -1

[for alternating, prev & curr they should be different.]

T.C → O(n*l)
S.C → O(1)

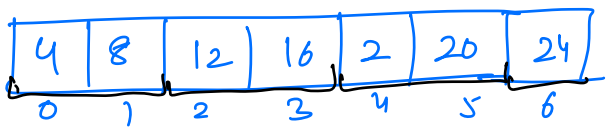
l = 2B+1

Even Sub-arrays.

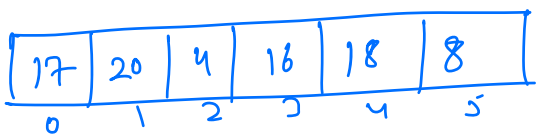
Q: Integer array : A

Can we divide the array into even length subarrays such that first & last elements of subarrays are even.

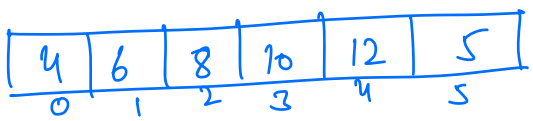
⇒ "YES" , "NO"

a: 

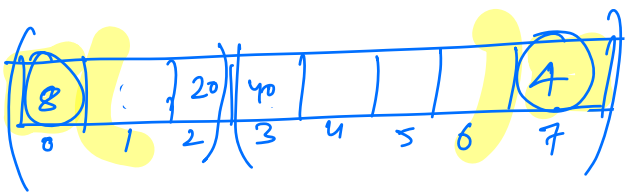
odd ⇒ "NO"

b: 

⇒ "NO"

c: 

⇒ "NO"

d: 

⇒ YES.

[len → even, first element & last element ⇒ even]

Good Sub-arrays

Q: Good sub-array

→ length → even, sum of all elements $< B$

→ length → odd, sum of all elements $> B$.

Count of good subarrays in array?

$B = 15$

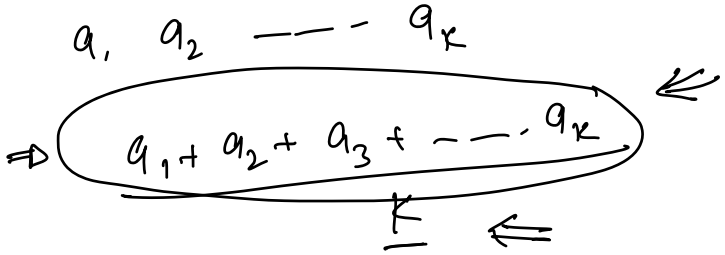
Eg: [6 4 3 7 8 1]
0 1 2 3 4 5 ans →

pseudo-code.

```
for( i = 0; i < N; i++){
    sum = 0;
    for( j = i; j < N; j++){
        sum += a[j];
        len = j - i + 1;
        if( len is even && sum < B) count++;
        if ( len is odd && sum > B) count++;
    }
}
```

T.C → $O(N^2)$
S.C → $O(1)$

Q.1 Sub-array of size - K with least average. //



[Sub array of size-K which is having the least sum.]

Doubts: A → [1, 2, 3, 4, 5]

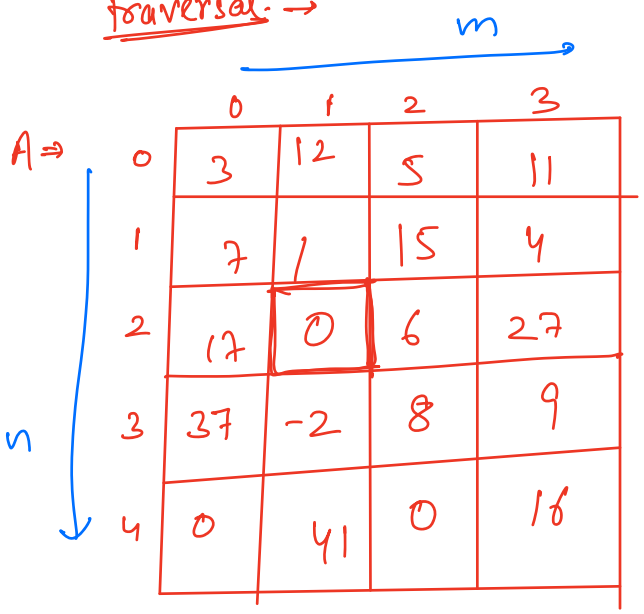
B → [2, 3]

5 4 3 2 1

[[4 5 1 2 3] [3 4 5 1 2]]

int [] [] ans = new int [B.length] [A.length]

traversal →

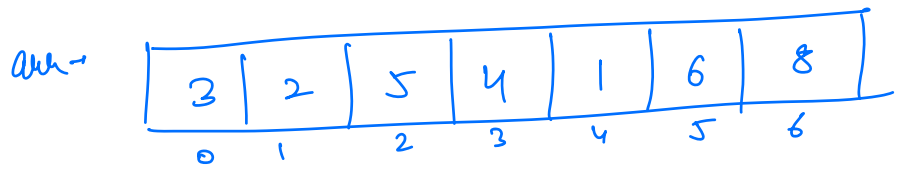


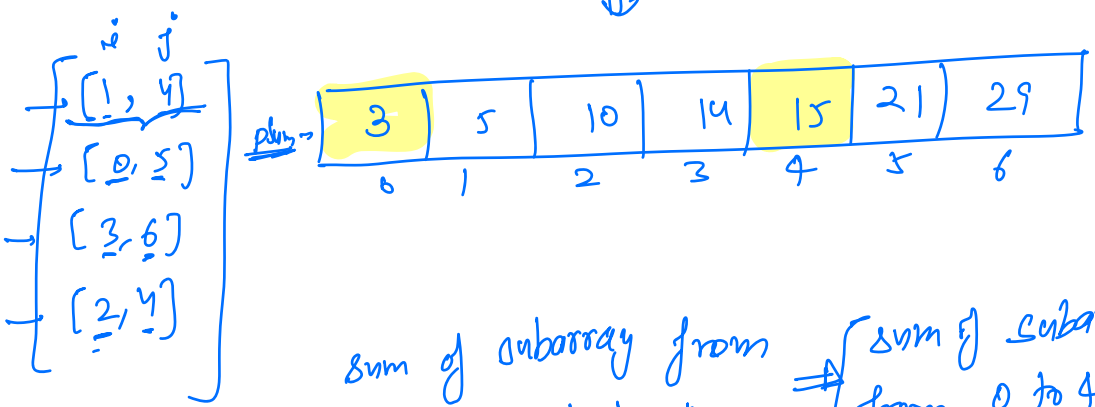
[i, j]

```
for(col = 0; col < m; col++) {
    arr[i][col] = 0;
}
for(row = 0; row < n; row++){
    arr[row][j] = 0;
}
```

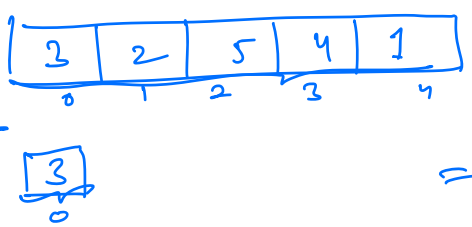
list 1 → [ ]

list 2 → [





sum of subarray from 1 to 4 = [sum of subarray from 0 to 4 − sum of subarray from 0 to 0]



= psum[j] − psum[i−1]

(i, j) ⇒ psum[j] − psum[i−1].