TODAY:

→ prefix sum

→ problems on prefix sum.

Given a array representing daily profit or loss from a particular stock. function that calculate total profit or loss in a given range of days.

→ Stock price = [-5, 10, 20, 40, 50, -10, 80, -90, -20, -10]
(1 2 3 4 5 6 7 8 9 10)

→ Queries [[start, end], [_ , _]]

start	end	Profit / loss
1	1	-5
1	2	-5 + 10 = 5
2	5	10+20+40+50 = 120.
8	10	-120
3	8	90.
Range → L	R	

sb [_ , _ , _ ,] and [_ , _ , _].

→ Given N elements and Q queries. for each query, calculate sum of all elements from L to R / S to E [0 based index].

A = { -3, 6, 2, 4, 5, 2, 8, -9, 3, 1 }
(0 1 2 3 4 5 6 7 8 9)

L	R	Sum
→ 4	8	9
→ 3	7	10.

[[1,2], [3,4], [5,6]]
(0 1 2)

pseudocode

```
function querySum ( A[], Q[][], querySize, size ) {
    for ( i → 0 to querySize ) { // this will iterate Q[][]
        L = Q[i][0], R = Q[i][1]
        sum = 0;
        for ( j → L to R ) {
            sum += A[j];
        }
        print(sum);
    }
}
```

(L = 2, R = 2, querySize = 2, size = 10)

A: { -3, 6, 2, 4, 5, 2, 8, -9, 3, 1 }

Q: [L R Z

0 [2, 4], → 11

1 [4, 6], → 15

2 [7, 8], → -6

3 [3, 7], → 10

4 [6, 9]. → 3

]

Q.length.

```
func querySum(A[], Q[][]) {                          → O(Q*N)
    int qlength = Q.length;
    int alength = A.length;

    for(i → 0 to qlength) {                          → Q
        L = Q[i][0];
        R = Q[i][1];
        sum = 0
        for(j → L to R) {                            → N
            sum = sum + A[j];
        }
        print(sum);
    }
}
```

	1	2	3 (L)	4	5	6	7	8	9 (R)	10
Cummulative sum →	2	8	14	29	31	49	65	79	88	97

Prefix sum

Sum 3-5 = sum[1-5] − sum[1-2]

31 − 8

65 − 49 = 16 //

Score in 7th ques = Sum[1-7] − Sum[1-6]

Score 6 − 10th = 97 − 31 = 66 // (L, R)

R − [L-2]

Score 10th = [10-10] = R − [L-1]

97 − 88 = 9 //. (L, R)

Score [3−6] = R − [L-1]

49 − 8 = 41 //. (L, R)

Score [4 - 9] = R − [L-1]

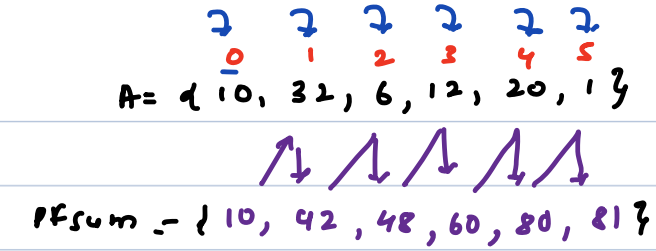
88 − 14 = 74 //. (L, R)

Prefix sum

	0	1	2	3	4
A = {	2	5	-1	7	1 }

	0	1	2	3	4
Pfsum = {	2	7	6	13	14 }

Pfsum[i]	cumulative sum [0-i].
0 →	2
1 →	2 + 5
2 →	2 + 5 - 1
3 →	2 + 5 - 1 + 7
4 →	2 + 5 - 1 + 7 + 1



pseudocode →

```
A[N], Pf[N].
for (i → 0 to N) {                    → N        O(N²)
    sum=0;
    for (j → 0 to i) {                → N
        Sum = sum + A[i]
    }
    Pfsum[i] = Sum;
}
```

Optimise... ???

Pf[0] = A[0]

Pf[1] = A[0] + A[1]

Pf[2] = A[0] + A[1] + A[2].

Pf[3] = A[0] + A[1] + A[2] + A[3]

Pf[3] = Pf[2] + A[3]

Pf[4] = Pf[3] + A[4].

Pf[5] = Pf[4] + A[5]

⋮

Pf[i] = Pf[i-1] + A[i].

pseudocode to calculate Pfsum

```
A[N], Pfsum[N], Pfsum[0] = A[0].
for(i → 1 to N){                          → N
    Pfsum[i] = Pfsum[i-1] + A[i];
                0          0-1 + A[0]
}
```

T.C : O(N)

optimised approach

→ stock prices = [-3, 6, 2, 4, 5, 2, 8, -9, 3, 1]
(0 1 2 3 4 5 6 7 8 9)

pfsum = [-3, 3, 5, 9, 14, 16, 24, 15, 18, 19]
(0 1 2 3 4 5 6 7 8 9)

Sum[L-R] = Pfsum[R] - Psum[L-1]

Sum(0-5) = Pfsum[5] - Psum[-1]. XX

if L==0
↳ = Pfsum[R].

```
function querySum(stock[], Q[][]){ TC: O(N+Q)
    n = stock.length; q.length = Q.length;
    Pfsum[n]; // create Pfsum array of size n.
    Pfsum[0] = A[0]
    for(i → 1 to N){                          → N
        Pfsum[i] = Pfsum[i-1] + A[i];
                    0          0-1 + A[0]
    }
    for(i → 0 to q.length){ // loop Q[][] → Q
        L = Q[i][0], R = Q[i][1];
        if(L==0){ print(Pfsum[R]}
        else{ print(Pfsum[R] - Psum[L-1]}
    }
}
```

10:31 pm
↓
10:40 pm

Rick & Morty

Sat :→ Tomo: 9pm

Q→ Given an array of N, & Q queries with s and e index for every query, return the sum of all even indexed elements. from s to e.

	0	1	2	3	4	5
A= {	2	3	1	6	6	5 }

query [
[1, 3] → 1 2 3 : × ✓ × : A[2] = 1 //
[2, 5] → 2 3 4 5 : ✓ × ✓ × : A[2] + A[4] = 5 //
[0, 4] → 0 1 2 3 4 : ✓ × ✓ × ✓ : A[0] + A[2] + A[4] = 7
[3, 3] → 3 : × = 0 ← //.
]

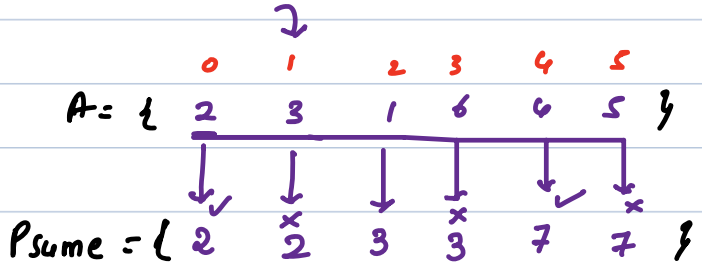
Brute force :- we will iterate every q and get s and e and from s and e check if index is even the add.

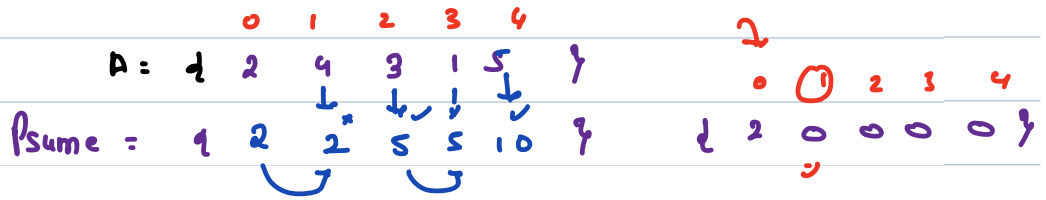
```
for (i → 0 to Q.length) {                    → Q
    s = Q[i][0], e = Q[i][1], sum = 0
    for (j → s to e) {                       → N
        if (j % 2 == 0) {
            sum += A[j];
        }
    }
    print(sum);
}
```

T.C: O(N*Q)

PRO TIP :. **Range sum → use Pfsum**

Prefix sum of Even elements






```
function evenSumQr( stock[], Q[][] ) {   TC: O(N+Q)
    n = stock.length; qlength = Q.length;
    Pfsum[n]; // create Pfsum array of size n.
    Pfsum[0] = A[0]
    for( i ⟶ 1 to N ) {
        if( i%2 == 0 ) {
            Psum[i] = Psum[i-1] + A[i];
        }
        else {
            Psum[i] = Psum[i-1];
        }
    }
    for( i ⟶ 0 to qlength ) {   // Loop Q[][] ⟶ Q
        S = Q[i][0], E = Q[i][1];
        if( S == 0 ) { print( Pfsum[E] ) }
        else { print( Pfsum[E] - Psum[S-1] ) }
    }
}
```

Another version : sum of odd elements from S - E.

Given an array of size N, count the number of special index in the array.

Note: Special Index :-

those indices after removing which,

Sum of all even index elements == Sum of all odd index elements.

	0	1	2	3	4	5
A[] : {	4	3	2	7	6	-2 }

i	A[i]	Se	So	Special Index
0	3 2 7 6 -2	8	8	→ ✓ c:0→1
1	4 2 7 6 -2	9	8	→ x
2	4 3 7 6 -2	9	9	→ ✓ c=1→2
3	4 3 2 6 -2	4	9	→ x
4	4 3 2 7 -2	4	10	→ x
5	4 3 2 7 6	12	10	→ x

Ans=2//.

	0	1	2	3	4
A= {	4	1	3	7	10 }

	0	1	2	3	Se	So
0	1	3	7	10	8	13
1	4	3	7	10	14	13
2	4	1	7	10	11	11
3	4	1	3	10	7	11
4	4	1	3	7	7	8

	0	1	2	3	4	5	6	7	8	9
A=	2	3	1	4	0	-1	2	-2	10	8

	0	1	2	3	4	5	6	7	8
	2	3	1	0	-1	2	-2	10	8

3

i = 3

i+1 N

0 ——— i-1 4 ——— 9

1 = 3 + sum of all even elmnt from 4–9.

i = 2

	0	1	2	3	4
A =	4	1	3	7	10
	4^0	1	7^2	10^3 ← odd	

0 — i-1 : Sum of all odd indices

i+1 ⟶ N : sum of all even indices.

	0	1	2	3	4	5	6	7	8	9
A =	2	3	1	4	0	-1	2	-2	10	8
	0	1	2	3	4	5	6	7	8	
	2	3	1	0	-1	2	-2	10	8	

0 — i-1 : Sum of all even indices

i+1 ⟶ n : Sum of all odd indices

2^0 i-1

	0	1	2	3	4	5	6	7	8	9
A =	2	3	1	4	0	-1	2	-2	10	8
	0	1	2	3	4	5	6	7	8	
	2	3	1	0	-1	2	-2	10	8	

0 — i-1 : Sum of all even indices

i+1 ⟶ n : Sum of all odd indices

optimised approach

1) calculate prefix sum of even indices
2) calculate prefix sum of odd indices

3) loop from i ⟶ 0 to N-1

evensum; count=0;

oddsum.

. .

evensum = Sum of even indices [0-i-1] +
Sum of odd indices. [i+1-n].

oddsum = Sum of odd indices [0-i-1] +
Sum of even indices [i+1-n].

```
if( oddsum == evensum ){
|     count++;
}
```

pseudocode

```
function count_special_index ( arr[], n ){
    Pfsum[ ] // calculate
    PSumo[ ] // calculate.
    count=0;
    for (i ⟶ 0 to n-1 ){
        if (i==0){
            // Todo 2 Hw.
        }
        else{
            // Pse[0→i-1]  Pso[i+1 → N]
            Se= Pse[i-1] + Pso[N] - Pso[i];
            So= Pso[i-1] + Pse[N] - Pse[i];
            // Pso[0→i-1] + Pse[i+1→N].
        }
        if (Se == So){
            count ++.
        }
    }
    print(count);
}
```

Next class

- → Carry forward Technique.
- → Subarrays
- → Problem on both