Ques

Given N elements and Q queries. For each query, calculate sum of all elements from L to R [0 based index].

0 1 2 3 4 5 6 7 8 9

A[ ] = [-3, 6, 2, 4, 5, 2, 8, -9, 3, 1]

L R

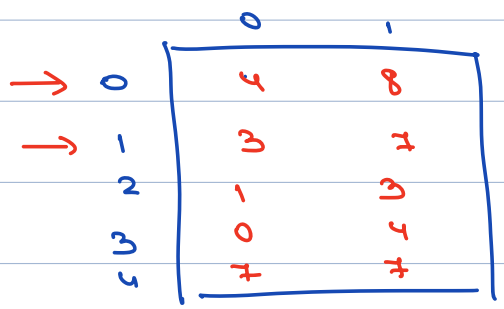
→ 4 8 → 9

→ 3 7 → 10

→ 1 3 → 12

→ 0 4 → 14

→ 7 7 → -9



← Brute force →

for every query travel from L to R.

```
Function querySum(Queries[][],Array[],querySize,size){
     for(i = 0; i < Queries.length; i++){  → O(n)
        L  = Queries[i][0]
        R = Queries[i][1]

        sum = 0
        for( j = L ; j <= R ; j++){  → n
           sum += Array[j]
        }
        print(sum)
     }
}
```

T.C → O(n*n)

S.C → O(1).

$1 <= N <= 10^5$

$1 <= Q <= 10^5$

The above code will throw TLE.

Ques.

Given the scores of the 10 overs of a cricket match

1 2 3 4 5 6 7 8 9 10

2, 8, 14, 29, 31, 49, 65, 79, 88, 97

How many runs were scored in just 7th over?

65 - 49 = 16 runs.

C_1 C_2 C_3 C_4 C_5 C_6 7 8 9 10

2, 8, 14, 29, 31, 49, 65, 79, 88, 97

↳ Cummulative Sum.

↓

Prefix Sum.

runs in

6th to 10th Over → 97 - 31 => 66.

10th Over → 97 - 88 => 9

3rd to 6th Over → 49 - 8 => 41

4th to 9th Over → 88 - 14 = 74.

Prefix Sum Array

pf[i] = Sum of all elements from 0 till ith idx.

	0	1	2	3	4
arr[] =	2	5	-1	7	1
pf[] =	2	7	6	13	14

	0	1	2	3	4	5
arr[] =	10	32	6	12	20	1
pf[] =	10	42	48	60	80	81 ✓

```
pf[N];
for (i=0; i<n; i++) {
    sum=0;
    for (j=0; j<=i; j++) {
        sum= sum+ A[j];
    }
    pf[i]= sum;
}
```

T.C → $O(n^2)$.

(0-3) (0-2)
pf[3] = pf[2] + arr[3];

(0-i) (0-(i-1))
pf[i] = pf[i-1] + arr[i];

```
pf[N];
pf[0] = arr[0];
for(i=1; i<n; i++) {
    pf[i]= pf[i-1]+ arr[i]
}
```

T.C → O(n)

	0	1	2	3	4	5
arr[] =	10	32	6	12	20	1
pf[] =	10	42	48	60	.	.

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

pf[] = [-3, 3, 5, 9, 14, 16, 24, 15, 18, 19]

L R

→ 4 8 → pf[8] - pf[3] = 18 - 9 = 9

→ 3 7 → pf[7] - pf[2] = 15 - 5 = 10

→ 1 3 → pf[3] - pf[0] = 9 - (-3) = 12

→ 0 4 → pf[4] = 14

→ 7 7 → pf[7] - pf[6] = 15 - 24 = -9

$$sum(l-r) = \begin{cases} pf[r] - pf[l-1] & (l != 0) \\ pf[r] & (l = 0) \end{cases}$$

```
Function querySum ( Queries[][], arr[] ) {
    // Creating pf array                    pf[N] → O(N)
    for ( i = 0; i < Queries.len; i++ ) {   → O(Q).
        L = Queries[i][0];
        R = Queries[i][1];
        if (L == 0) {
            sum = pf[R]
        }
    }
```

```
        else {
            sum = pf[R] - pf[L-1];
        }
        print(sum);
    }
}
```

T.C → O(N + Q)

S.C → O(N).

$1 <= N <= 10^5$

$1 <= Q <= 10^5$

$(10^5 + 10^5)$

2×10^5.

We can optimize the S.C as well :-

↳ using the input array as prefix array.

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

```
Function prefixSumArrayInplace(Array[],size){
    for(i = 1; i < size; i++){
        Array[i] = Array[i-1] + Array[i];
    }
}
```

Ques

Given an array of size N and Q queries with start (s) and end (e) index. For every query, return the sum of all **even indexed elements** from **s to e**.

```
                 0  1  2  3  4  5
arr[] = { 2, 3, 1, 6, 4, 5 }
```

Queries

l	r	sum
1	3	1
2	5	5
0	4	7
3	3	0

Brute force :-

for every query, travel l to r
& add only even idx elements.

T.C → O(N * Q)

Optimized idea :-

```
                 0  1  2  3  4  5
arr[] = { 2, 3, 1, 6, 4, 5 }
pfe[] =   2  2  3  3  7  7
```

Queries

l	r	sum
1	3	1
2	5	5
0	4	7
3	3	0

pfe[0] = arr[0]
i is even,
pfe[i] = pfe[i-1] + arr[i]
if i is odd,
pfe[i] = pfe[i-1];

```
                0  1  2  3  4
arr[] = [ 2  4  1  3  5 ]
pfe[] =   2  2  3  3  8
```

↳ sum of all even idx elements till me.

	0	1	2	3	4	5
arr[] = {	2,	3,	1,	6,	4,	5 }
pfe[] =	2	2	3	3	7	7

Queries

L	R	sum
1	3	pfe[3] - pfe[0] = 3 - 2 => 1
2	5	pfe[5] - pfe[1] = 7 - 2 => 5
0	4	pfe[4] = 7
3	3	0

```
pse[n];

pse[0] = arr[0];

for(i=1; i<n; i++) {
    if(i%2==0) {
        pse[i] = pse[i-1] + arr[i];
    }
    else {
        pse[i] = pse[i-1];
    }
}

for(i=0; i<queries.size; i++) {
    L = queries[i][0];
    R = queries[i][1];
    if(L==0) {
        sum = pfe[R];
    }
    else {
        sum = pfe[R] - pfe[L-1];
    }
    print(sum);
}
```

T.C -> O(N + Q)

S.C -> O(N).

for Odd Idxes :-

```
          0  1  2  3  4  5
arr[] = { 2, 3, 1, 6, 4, 5 }
pfo[] = { 0, 3, 3, 9, 9, 14 }
```

```
pfo[0] = 0;
if i is odd;
    pfo[i] = pfo[i-1] + arr[i]
else;
    pfo[i] = pfo[i-1];
```

Ques) Special Idx :-

Given an array of size N, count the number of special index in the array.
Note: Special Indices are those after removing which, sum of all **EVEN** indexed elements is equal to sum of all **ODD** indexed elements.

```
          0  1  2  3  4  5
arr[] = { 4, 3, 2, 7, 6, -2 }
```

i	0 1 2 3 4	se	so
0	{ 3, 2, 7, 6, -2 }	8	8
1	{ 4, 2, 7, 6, -2 }	9	8
2	{ 4, 3, 7, 6, -2 }	9	9
3	{ 4, 3, 2, 6, -2 }	4	9
4	{ 4, 3, 2, 7, -2 }	4	10
5	{ 4, 3, 2, 7, 6 }	12	10

Ans → 2.

```
se = so(1 to n);
so = se(1 to n);
```

A[] = { 4, 1, X, 7, 10 } ⇒ $so[0-1] + se[3-4]$ ⇒ 1 + 10 ⇒ 11

Sum of odd, after delete idx 2,

A[] = { 4, 1, 7, 10 }

What will be the sum of elements at ODD indices in the resulting array after removal of index 3 ?

A[ ] = { 2, 3, 1, X, 0, -1, 2, -2, 10, 8 }

↳ $so[0-2] + se[4-9]$

3 + 12 ⇒ 15.

A[] = { 2, 3, 1, 0, -1, 2, -2, 10, 8 }

→ Sum of Even Idx, after deleting idx 3,

[2, 3, 1, X, 0, -1, 2, -2, 10, 8]

$se[0-2] + so[4-9]$

3 + 5 ⇒ 8.

$so = so[0-(i-1)] + se[(i+1)-(m-1)]$ ✓

$se = se[0-(i-1)] + so[(i+1)-(m-1)]$ ✓

$so = pfo[i-1] + pfe[m-1] - pfe[i]$;

$se = pfe[i-1] + pfo[m-1] - pfo[i]$;

So(l-r)
= Pfo[r] l==0
= Pfo[r] l!=0
- Pfo[l-1]

Se(l-r), = Pfe[r] l==0
Pfe[r] - Pfe[l-1] l≠0

final Code :-

int Pfe[n];
int Pfo[n];
int count=0;
for(i=0; i<n; i++) {
// delete ith idx :-
So = 0;
Se = 0;
if(i=0) {
So = Se(i+1 to n-1)
Se = So(i+1 to n-1);
} else {
So = Pfo[i-1] + Pfe[n-1] - Pfe[i]
Se = Pfe[i-1] + Pfo[n-1] - Pfo[i];
}
} Todo.

Pfe[n-1] - Pfe[i]

Pfo[n-1] - Pfo[i].

```
    }
    if (so == se) {
        cont++;
    }
}
return cont;
```

T.C → O(n)

S.C → O(1)

1. Carry forward. –
2. Subarrays –

Problem Solving

Applications → Interview Problems.