Today's Content

→ Prefix Sum introduction

→ Problems based on prefix number.

Q. Given an array(N), and Q queries.

For each query, given L & R. we need to calculate sum of elements from [L, R]

	0	1	2	3	4	5	6	7	8	9
arr[10] =	-3	6	2	4	5	2	8	-9	3	1

Q = 6.

L	R	sum
4	8	9
3	7	10
1	3	12
0	4	14
6	9	3
7	7	-9

→ For every query iterate [L, R] & get sum.

= 6

Q, arr

```
while(Q--)
{
    int L, R.  read L, R.
    sum = 0
    for(i=L; i<=R; i++)
    { sum = sum + arr[i]
    print(sum)
}
```

TC: O(N×Q)

SC: O(1)

Q. Given Indian Team scoreboard, for last 10 overs.

41	42	43	44	45	46	47	48	49	50
288	312	330	349	360	383	394	406	436	439

Q. Runs scored in last 5 overs?

[46-50]

$439 - 360 = 79$

R[50] - R[45]

Q. Run scored in 50th over?

[50-50]

$439 - 436 = 3$.

R[50] - R[49]

Q. Run scored in 49th over?

[49-49]

$436 - 406 = 30$

R[49] - R[48]

Q Run scored in 42-47th over?

[42,47]

R[47] - R[41]

	0	1	2	3	4	5	6	7	8	9
arr[10] =	-3	6	2	4	5	2	8	-9	3	1
PF[10] =	-3	3	5	9	14	16	24	15	18	19

sum[0,7] = sum[0-1] + sum[2-7]

sum[2,7] = sum[0-7] - sum[0-1]

= 15 - 3 = 12

Array → Given. Cummulative sum = Prefix array

PF[i] = sum of all elements [0, i]

Sum[4, 8] = PF[8] − PF[3]

Sum[i, j] = PF[j] − PF[i−1]

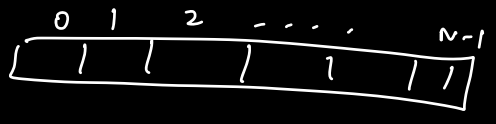
Sum[L, R] = PF[R] − PF[L−1]

Sum[L, R] → PF[R] if L = 0

PF[R] − PF[L−1] if L > 0

// Construct PF sum.

given arr[N]



PF[0] = arr[0]

PF[1] = arr[0] + arr[1]

PF[2] = arr[0] + arr[1] + arr[2]

arr[0] + arr[1] = PF[1]

PF[i] = arr[0] + a[1] + arr[2] + ... a r[i−1] + a[i]

arr[0] + a[1] + arr[2] + ... a r[i−1] = PF[i−1]

PF[i] = PF[i-1] + arr[i]

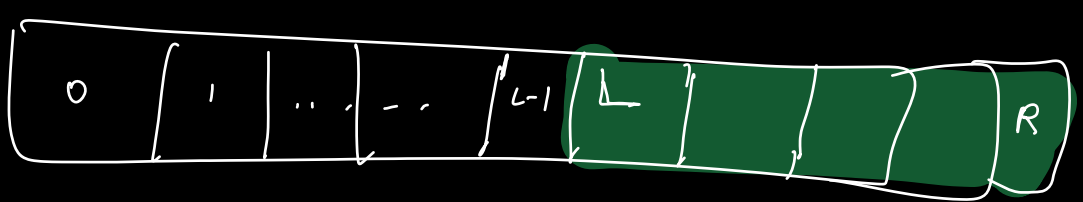
4N + 12

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

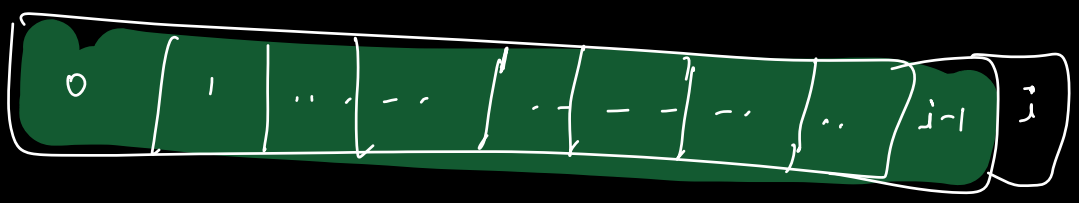
N

TC : O(N+Q)
SC : O(N)

```
int Q.   read Q.
while(Q--)
{   int L,R.   read L,R.
    if(L==0)   print(PF[R])
    else       print(PF[R] - PF[L-1])
```



PF[R] - PF[L-1]



$PF[i] = PF[i-1] + arr[i]$

Direct
Amazon
microsoft

Q. Given an array. Return true if exists an equilibrium index.

EI : index of which

sum of left side = sum of right side

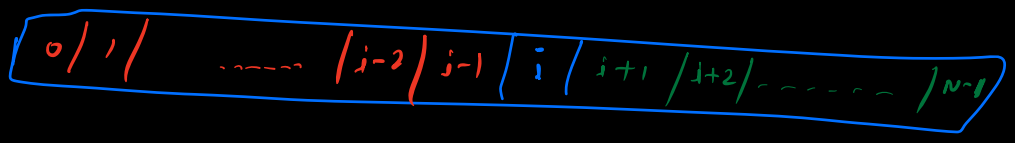
for index 0 $S_L = 0$

for idex n-1 $S_R = 0$

x	x	x	x		
0	1	2	3	4	5
1	2	3	4	8	10

ans = True

Approach for every index, check if is a EI



For checking i

sum [0, i-1] == sum [i+1, N-1]
(L, R)

PF[i-1] PF[N-1] - PF[i+1-1]

PF[i-1] == PF[N-1] - PF[i]

// construct PF

```
for( i=0 ; i<N ; i++)
{
    if ( i==0 )  SL = 0
    else         SL = PF[i-1]

    if ( i == N-1 ) SR = 0
    else            SR = PF[N-1] - PF[i]

    if ( SL == SR )  return True
}

return False
```

TC : O(N)

SC : O(N)

Q. Given an array, Calculate Q queries.

L, R, O ⇒ sum of all odd-indexed elements

L, R, E ⇒ sum of all even-indexed elements.

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

Q = 2

L	R	o/E
3	6	O
1	5	E

arr[3] + arr[5] = -1 + 8 = 7

arr[2] + arr[4] = 1 + 0 = 1

Let try to create 2 prefix sum.

$PS_e[i]$ → sum of even indexed elements from [0, i]

$PS_o[i]$ → sum of odd indexed elements from [0, i]

	0	1	2	3	4
Arr =	2	4	3	1	5
PF_o	0	4	4	5	5
PF_e	2	2	5	5	10

PFO[0] = 0

```
for ( j = 1 ; j < N ; j++)
{
    if ( i%2 == 0 )   PFo[i] = PFo[i-1]
    else              PFo[i] = PFo[i-1] + arr[i]
```

```
pFE[0] = arr[0]

for( j=1 ; j<N ; j++)
{
    if( j%2 == 1 )    pFE[j] = pFE[j-1]
    else              pFE[j] = pFE[j-1] + arr[j]
}
```

TC: O(N+Q)

SC: O(N)

```
while(Q--)
{
    L, R, O
        ↓
        pFO[R] - pFO[L-1]

    L, R, E
        ↓
        pFE[R] - pFE[L-1].
}
```

Direct Codenation

Google

JP morgan.

Q, Given an array. Count the no of special index in the array.

Special index = An index is SI, if after removing this index

Sum of all odd-indexed elements = Sum of all even-indexed elements.

	0	1	2	3	4	5
A =	4	3	2	7	6	-2

j = 0	3	2	7	6	-2
j = 1	4	2	7	6	-2
j = 2	4	3	7	6	-2

$S_O 8 = S_E(8)$ ✓

$S_O = 8$, $S_E = 9$ ✗

$S_O = 9$, $S_E = 9$ ✓

j = 3

j = 4

j = 5

	0	1	2	3	4	5
	4	1	5	3	7	10

	0	1	2	3	4
	4	1	3	7	10

$S_{odd} = 14$

$S_{odd} = 8$

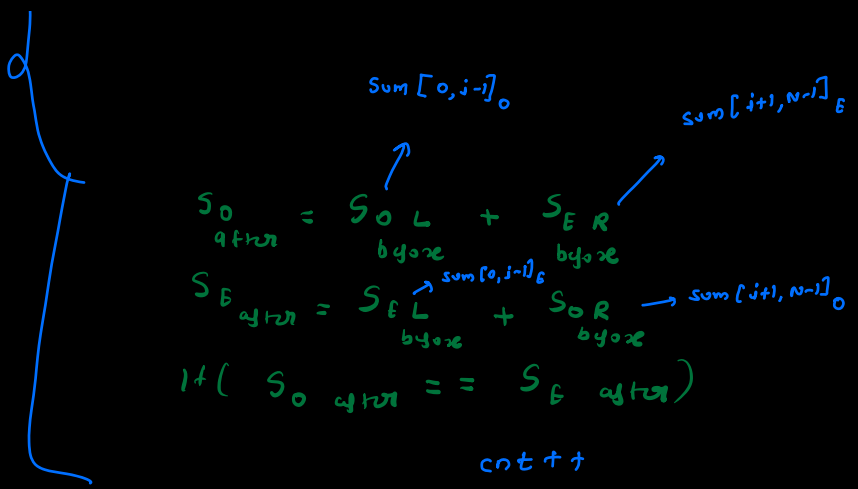
	0	1	2	3	4	5
	4	1	5	3	7	10

Sum of all O after deleting = Sum O in left before + Sum E in Right before

arr = [ ]

PF_O PF_E Sum_O Sum_E

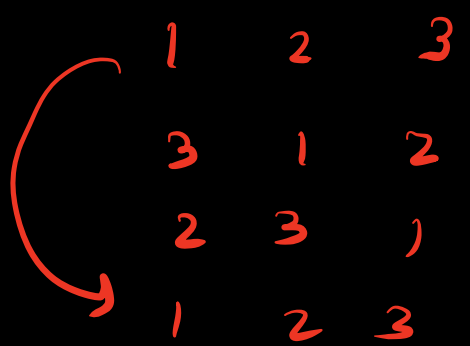
```
for ( j = 0 ; j < n ; j++ )
{
    // i is EF or not
```


Sum [0,i-1]o
Sum [i+1,N-1]E
So after = So L before + SE R before
SE after = SE L before + So R before
Sum [0,i-1]E
Sum [i+1,N-1]o
if (So after == SE after)
cnt++

b = b % N

16 % 3 = 1

N = 3


1 2 3
3 1 2
2 3 1
1 2 3