Today's Quote →

Today's content.

- prefix sum
- problems on prefix sum.

Q) Given N array elements and Q (no. of queries) queries. for each query-
- calculate sum of all elements in range - [L, R]

Note → L and R are indices such that L <= R. $1 \le N, Q \le 10^5$

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

Queries - S

L	R	ans.
4	8	9
3	7	10
1	3	12
0	4	14
7	7	-9

idea → for every query, calculate the sum. [iterate from l to r].

pseudo-code -

```
void fun( arr , N){
    q → take input [no. of queries]
    while ( q > 0) {
        q -= 1
        l , r → take input
        sum → 0
        for( i = l ; i <= r ; i++){
            sum += arr[i]
        }
        print( sum) ;
    }
}
```

T.C → O(N*Q)
S.C → O(1)

{ T.L.E }

Q) Given Indian Cricket Team scores for first 10 overs of batting. After every over, total score is given as:

Overs:	1	2	3	4	5	6	7	8	9	10
Score-board:	2	8	14	29	31	49	65	79	88	97

cumulative sum

Total runs scored in last over: $97 - 88 = 9$

score[10] - score[9]

Q) Total runs scored in 7th over: $65 - 49 = 16$

score[7] - score[6]

Q) Total runs scored in overs 6th to 10th: $97 - 31 = 66$

score[10] - score[6-1]

Q) Total runs scored in overs 3rd to 6th: $49 - 8 = 41$

score[6] - score[3-1]

Total runs from ith over to jth over → score[j] - score[i-1]

idea. ⇒ store cumilative sum / prefix sum.

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

Queries - s ans.

L	R			
4	8	→	psum[8] - psum[4-1]	= 18 - 9 = 9
3	7	→	psum[7] - psum[3-1]	= 15 - 5 = 10
1	3	→	psum[3] - psum[1-1]	= 9 - (-3) = 12
7	7	→	psum[7] - psum[7-1]	= 15 - (24) = -9
0	4	→	psum[4]	= 14.

[l r → psum[r] - psum[l-1]]

[psum[i] → Sum of all the elements from index 0 to i.]

How to construct prefix array.

{ 3, -2, 4, 5, 6 }.
0 1 2 3 4

$pSum[0] = arr[0] = 3$

$pSum[1] = arr[0] + arr[1]$

$pSum[1] = pSum[0] + arr[1]$

$pSum[2] = arr[0] + arr[1] + arr[2]$

$pSum[2] = pSum[1] + arr[2]$

$pSum[3] = arr[0] + arr[1] + arr[2] + arr[3]$

$pSum[3] = pSum[2] + arr[3]$

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

pseudo-code.

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

T.C → O(N)

pseudo. code for Q1

```
void fun ( arr, N) {
    psum[N];
    psum[0] = arr[0]
    for( i = 1 ; i < N ; i++){
        psum[i] = psum[i-1] + arr[i]
    }                                          } N

    q → take input
    while ( q > 0) {
        q -= 1
        l , r → take input.
        if (l == 0) {
            print( psum[r] );
        else {
            print ( psum[r] - psum[l-1] );
        }
    }                                          } Q
}
```

T.C → O(N+Q)

S.C → O(N)

Can we modify the array ?

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

||

arr[10] : [-3 3 5 9 14 16 24 15 18 19]

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

Advantage : SC is optimised.

Dis-advantage : initial elements will be lost.

[Break 8:10 → 8:20]

Equilibrium Index.

Amazon Hike Adobe

Q) Given N array elements, count no of equilibrium index.

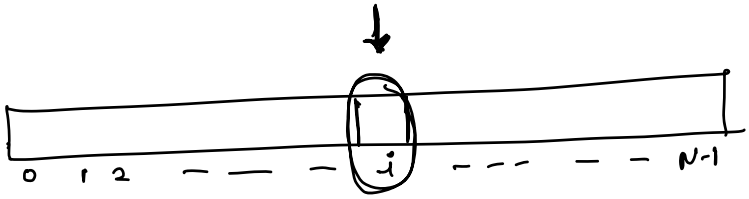
An index i is said to be equilibrium index if :

Sum of all elements on left of ith index = Sum of all elements on right of ith-index

Note: if i == 0, leftSum = 0
if i == N-1, rightSum = 0

sum[i, j] = pSum[j] - pSum[i-1]

= pSum[N-1] - pSum[i+j]



sum[0, i-1] = sum[i+1, N-1]

pSum[i-1] = pSum[N-1] - pSum[i]

Eg: arr → { -3 2 4 -1 }
0 1 2 3

leftSum → 0 -3 -1 3

rightSum → 5 3 -1 0

[ans = 1]
↑
count of equilibrium index.

Quiz. arr → [-7 1 5 2 -4 3 0]
0 1 2 3 4 5 6

options.
2/3/4/5

leftsum: 0 -7 -6 -1 1 -3 0

rightsum: 7 6 1 -1 3 0 0

logic and pseudo code →

Use-prefix Sum.

{ pSum[i-1] == pSum[N-1] - pSum[i] }

check this for every index.

pseudo-code.

```
int countEqIndex(arr, N){
    pSum[N];
    pSum[0] = arr[0]
    for(i=1; i<N; i++){
        pSum[i] = pSum[i-1] + arr[i]
    }                                   N
    count = 0
    for(i=0; i<N; i++){                 <- todo- when i=0.
        if(pSum[i-1] == pSum[N-1] - pSum[i]){
            count += 1
        }                               N
    }
    return count;
}
```

T.C → O(N)
S.C → O(N)

Q) Given N array elements and Q queries.
For each query l to r. find count of even numbers in given range.

Eg: arr[10] : { 2 4 3 7 9 8 6 5 4 9 }
0 1 2 3 4 5 6 7 8 9

Queries: 3

l	r	ans.
4	8	3
3	9	3
0	4	2.

Brute Force

For every query, iterate from l to r and find the count of even no's.

T.C → O(N*Q)
S.C → O(1)

pseudo-code.

```
void fun ( arr, N ) {
    q → take input.
    while ( q > 0 ) {
        q -= 1
        l, r → take input, count = 0
        for( i = l ; i <= r ; i++){
            if (arr[i] % 2 == 0) { count += 1 }
        }
        print (count);
    }
}
```

Optimisation :

arr[10] : { 2 4 3 7 9 8 6 5 4 9 }
0 1 2 3 4 5 6 7 8 9

even → 1
odd → 0

arr[10] : { 1 1 0 0 0 1 1 0 1 0 }
0 1 2 3 4 5 6 7 8 9

psum : { 1 2 2 2 2 3 4 4 5 5 }
0 1 2 3 4 5 6 7 8 9

[Count of even nos from l to r = psum[r] - psum[l-1]]

l = 0, r = 4 → psum[4] - psum[0-1] (0)

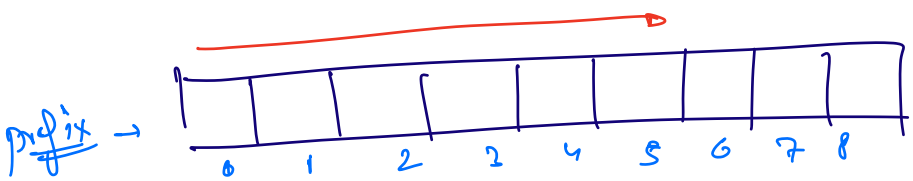
pseudo-code.

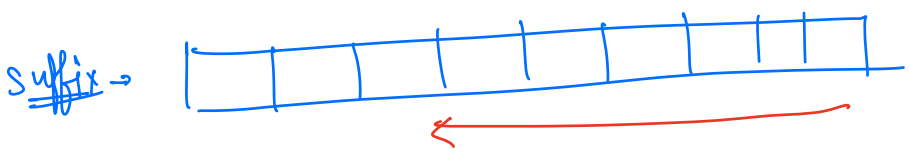
```
void fun(arr, N){
    for(i = 0; i < N; i++){
        if(arr[i] % 2 == 0){
            arr[i] = 1
        }
        else{
            arr[i] = 0
        }
    }
    for(i = 1; i < N; i++){
        arr[i] = arr[i-1] + arr[i]
    }
}
```

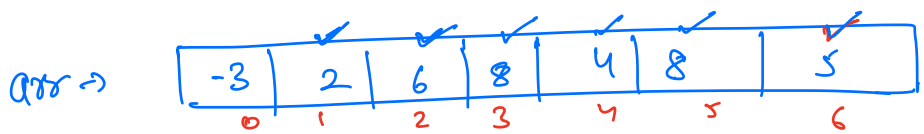
```
q → take input
while ( q > 0 ) {
    q -= 1 ;
    l, r → take input
    if (l == 0) { print ( arr[r] )
    else { print ( arr[r] - arr[l-1]);
}
}
```

T.C → O(N+Q)
S.C → O(1)

Doubts:





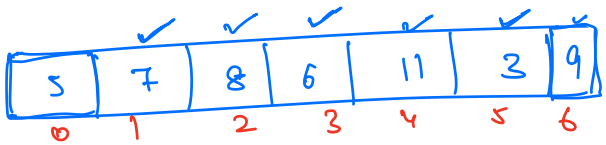


max = ~~-3~~ ~~2~~ ~~6~~ ~~8~~

countofmax = ~~1~~ ~~1~~ ~~1~~ ~~1~~ 2

```
if ( arr[i] > max){
        max = arr[i]
        countofMax = 1
}
else if (arr[i] == max){
        countofMax += 1
}
```

arr →



ans → 9.

max → ~~5~~ ~~7~~ ~~8~~ 11

smax → ~~5~~ ~~7~~ ~~8~~ 9

5

max → 5

smax → -1

```
if (arr[i] > max){
        smax = max
        max = arr[i]
}
else if ( arr[i] > smax){
        smax = arr[i]
}
```

arr → [-7 | 1 | 5 | 2 | -4 | 3 | 0]
indices: 0 1 2 3 4 5 6

psum → [-7 | -6 | -1 | 1 | -3 | 0 | 0]

```
int countEqIndex(arr, N){
    psum[N];
    psum[0] = arr[0]
    for(i = 1; i < N; i++){
        psum[i] = psum[i-1] + arr[i]
    }                                   N
    count = 0
    for(i = 0; i < N; i++){
        if(psum[i-1] == psum[N-1] - psum[i]){
            count += 1
        }                               N
    }
    return count;
}
```