Today's Quote →

Today's content:

- prefix sum
- problems on prefix sum.

For python students →

- Watch recordings of previous two classes.
- for any doubt
  - Ask peers
  - Ask TA's
  - Ping me on slack

5, 7, 9
M, W, F → 5 PM - 6 PM IST.

- As of now, focus on current content.
  If you only think about last classes, then certainly today's content will get affected.

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 - 6

L	R		ans.
4	8	:	9
3	7	:	10
1	3	:	12
0	4	:	14
6	9	:	3
7	7	:	-9

for every query, calculate the sum;
⇓
iterate from l to r & calculate sum.

pseudo. code.

```
void func (arr) {
    q → take input of queries.
    while( q > 0) {
        q -= 1;
        l, r → take input, sum=0
        for(i → l to r) {
            sum += arr[i]
        }
        print (sum)
    }
}
```

T.L → 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 [10th 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 cummilative sum / prefix sum.

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

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

Queries - 6

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.
0	4	:	pSum[4] = 14.

{ pSum[i] = Sum of all elements from [0 to i] }

How to construct prefix array.

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

$psum[0] = arr[0] = 3$

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

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

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

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

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

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

$\vdots$

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

pseudo-code →

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

⇒ TC → O(n)

pseudo. code for Q1

$l <= r$.

```
void fun ( arr ) {
    psum[n];
    psum[0] = arr[0];
    for ( i = 1 to n-1 ) {
        psum[i] = psum[i-1] + arr[i]
    }                                   O(n)
    q = take input for queries.
    while ( q > 0 ) {                   O(q)
        l, r → take input.
        if ( l == 0 )
            print ( prefixSum[r] )
        else
            print ( prefixSum[r] - prefixSum[l-1] )
        q -= 1 ;
    }
}
```

T.C → O(N+Q)

S.C → O(N)

Code →

Java

```java
public static void rangeQueries(int[] arr){
    int n = arr.length;
    int[] prefixSum = new int[n];
    prefixSum[0] = arr[0];
    for(int i = 1; i < arr.length; i++){
        prefixSum[i] = prefixSum[i - 1] + arr[i];
    }

    int q = scn.nextInt();
    while(q > 0){
        q -= 1;
        int l = scn.nextInt();
        int r = scn.nextInt();
        if(l == 0){
            System.out.println(prefixSum[r]);
        }else{
            System.out.println(prefixSum[r] - prefixSum[l-1]);
        }
    }
}
```

Python

```python
def rangeQuery(arr):
    n = len(arr)

    # generating the prefix sum array
    prefix_sum = [0] * n
    prefix_sum[0] = arr[0]
    for i in range(n):
        prefix_sum[i] = prefix_sum[i - 1] + arr[i]

    # no. of queries
    q = int(input())

    while(q > 0):
        q -= 1
        l, r = map(int, input.split())

        if(l == 0):
            print(prefix_sum[r])
        else:
            prefix_sum[r] - prefix_sum[l - 1];
```

Can we modify the array ?

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

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

```
for( i = 1 to n-1) {
    arr[i] = arr[i-1] + arr[i]
                        sum [0,i]
}
```

Break till 10:30

Advantage : S.C is optimised.

Dis-advantage : Initial elements are lost.

Equilibrium Index.

Amazon Hike Adobe

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

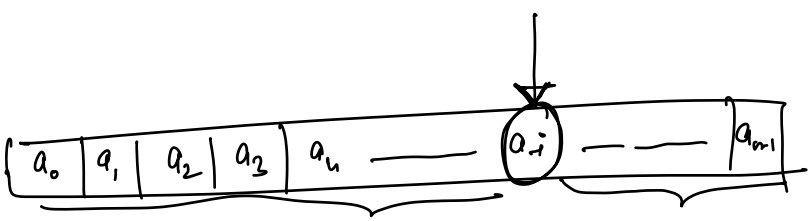
An index i is said to be equilibrium index if :

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

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

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

pSum[i] → sum[0 → i].



psum[i] - psum[i-1]
||
psum[n-1] - psum[i+1-1]

Sum[0 → i-1] == sum[i+1 → n-1]

⇓

[psum[i-1] = psum[n-1] - psum[i]]

Eg: arr[4] : { -3 2 4 -1 }
0 1 2 3

	0	1	2	3
leftSum :	0	-3	-1	3
rightSum :	5	3	-1	0

Quiz. Which one is eq. index. options → 2 | 3 | 4 | 5 (ans. 3)

Eg:

arr[7] : {	-7	1	5	2	-4	3	0 }	
	0	1	2	3	4	5	6	
leftSum :	0	-7	-6	-1	1	-3	0	
rightSum :	7	6	1	-1	3	0	0	

logic and pseudo code →

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

pseudo-code:

```
void countEqui ( arr ) {
    psum[n];
    psum[0] = arr[0];
    for ( i = 1 to n-1 ) {
        psum[i] = psum[i-1] + arr[i]
    }                                   ] O(N)
    count = 0
    for( i -> 0 to n-1 ) {
        if ( psum[i-1] == psum[n-1] - psum[i] )   ] O(N)
            count += 1
    }
    return count;
}
```

T.C → O(N), S.C → O(N)

Code →

Java

```java
public static void countEquilibrium(int[] arr){
    int n = arr.length;
    int[] prefixSum = new int[n];
    prefixSum[0] = arr[0];
    for(int i = 1; i < n; i++){
        prefixSum[i] = prefixSum[i - 1] + arr[i];
    }

    int count = 0;
    for(int i = 0 ; i < n; i++){
        int leftSum, rightSum;
        if(i == 0){
            leftSum = 0;
        }else{
            leftSum = prefixSum[i - 1];
        }
        rightSum = prefixSum[n-1] - prefixSum[i];
        if(leftSum == rightSum){
            count++;
        }
    }
    System.out.println(count);
}
```

Python

```python
def count_equilibrium(arr):
    n = len(arr)

    # generating the prefix sum array
    prefix_sum = [0] * n
    prefix_sum[0] = arr[0]
    for i in range(n):
        prefix_sum[i] = prefix_sum[i - 1] + arr[i]

    count = 0
    for i in range(n):
        if i == 0:
            left_sum = 0
        else:
            left_sum = prefix_sum[i-1]

        right_sum = prefix_sum[n-1]-prefix_sum[i]
        if left_sum == right_sum:
            count += 1

    print(count)
```

Q) Given N array elements and Q queries.

For each query l to r. find no. 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.

Pseudo-code →

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

T.C → O(N*Q)
S.C → O(1)

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

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

even no.'s from l to r. ⇒ psum[r] - psum[l-1].

4 to 8
0 to 4 psum[4] - psum[-1]

pseudo-code →

```
void fun( arr ){
    q → take input   // psum[] array.
    while( q > 0 ){
        q -= 1
        l, r → take i/p
        if( l == 0 ) print( arr[r] )
        else print( arr[r] - arr[l-1] )
    }
}
```

T.C → O(N+Q) , S.C → O(1)

→ sum [0 – i] , sum [i to n-1] , sum [i, j]

prefix Sum