Todays Content

1. PfSum Intro
2. Range sum queries

Thursday Class

1. leaders
2. Count Triplets
3. 0/1 prefixSum

Friday class

1. Quick C++ Transition
2. Vector in C++
3. Pass by value & Pass by reference
4. Vector< Vector<>> or Array of vector.
5. String in C++

Given ar[] & s_0, e_0 calculate sum of all elements from s→e:

Ex:

	0	1	2	3	4	5	6	7	8	9
ar[10] = {	3	4	6	8	9	10	2	7	4	10 }

s=2 e=7 : ans=42

```
int sum(int ar[], int s, int e){ TC: O(N) SC: O(1)
    int ans=0;
    for(int i=s; i<=e; i++){
        ans = ans + ar[i];
    }
    return ans;
}
```

Q1: Given ar(N) elements & Qmat[Q][2]

In Qmat matrix, we have Q: rows & 2: columns

Each row in Qmat represents a query.

0th col in row represents : start point of query → s = Qmat[i][0]

1st col in row represents : end point of query → e = Qmat[i][1]

For every query calculate sum of elements from index s..e in ar[] & print

Constraints:

$1 <= N <= 10^5$

$1 <= ar[i] <= 10^9$

$1 <= Q <= 10^5$

$0 <= s <= e < N.$

Sum = $\{1..10^{14}\}$ >> int rage

Min: 1 Max = $10^9 * 10^5 = 10^{14}$

ar[1] : {1} ar[10^5] : $\{\underline{10^9\ 10^9\ 10^9 .. 10^9}\}$

Ex:

	0	1	2	3	4	5	6	7	8	9
ar[10] =	{-3	6	2	4	5	2	8	-9	3	1}

Qmat[6][2]

	0:s	1:e	Output
→ 0	4	8	9
→ 1	3	7	10
→ 2	1	3	12
→ 3	7	7	-9
→ 4	3	6	19
→ 5	0	4	14

Idea1:

For every Query:

Iterate from s..e calculate sum & print it

Expected TC: O(Q*N)

	s	e
0th:	Qmat[0][0]	Qmat[0][1]
1st:	Qmat[1][0]:3	Qmat[1][1]:7
2nd:	Qmat[2][0]:1	Qmat[2][1]:3
ith:	Qmat[i][0]:s	Qmat[i][1]:e

```
void RangeSum(int ar[], int N, int Qmat[][], int Q){

    for(int i=0; i<Q; i++){
        int s = Qmat[i][0], e = Qmat[i][1];
        long sum = 0; // sum datatype should be long.
        for(int j=s; j<=e; j++){
            sum = sum+ar[j];
        }
        print(sum);
    }
}
```

Calculated TC: $O(Q*N)$ SC: $O(1)$

↓

$1 <= N <= 10^5$

$1 <= Q <= 10^5$ $10^5 * 10^5 = 10^{10} >> 10^8$ TLE.

Optimization Idea:

Say we are given csk cricket scores for first 10 overs of batting. After every over total score is given.

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

Q1: Total runs scored in 10th over = scre[10] - scre[9] = 9

Q2: Total runs scored in 7th over = scre[7] - scre[6] = 16

Q3: Total runs scored in 6th - 10th over = scre[10] - scre[5] = 66

Q4: Total runs scored in 3th - 6th over = scre[6] - scre[2] = 41

Q5: Total runs scored in 4th - 9th over = scre[9] - scre[3] = 74

Con: Total runs scored in ith - jth over = scre[j] - scre[i-1]

Obs: Total sum till that point = Cummulative sum.
Cummulative sum calculate from start = prefix Sum
prefix Sum = Total sum from 0 till that index i.
Storing prefix Sum value in array is psum[i]

Idea: Create pf[] to optimize

1. pf[N]
2. pf[i] = Sum of all elements [0.. i].

Ex:

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

Output: Ans using Psum[]

	0:s	1:e	pSum[e] - pSum[s-1]
0	4	8	→ pSum[8] - pSum[3] = 18 - 9 = 9
1	3	7	→ pSum[7] - pSum[2] = 15 - 5 = 10
2	1	3	→ pSum[3] - pSum[0] = 9 - (-3) = 12
3	7	7	→ pSum[7] - pSum[6] = 15 - 24 = -9
4	3	6	→ pSum[6] - pSum[2] = 24 - 5 = 1
5	0	4	→ pSum[4] - pSum[-1] // Error

→ sum[0..4] = pSum[4].

Query:

```
[s..e] = if( s==0){ // [0..e]
             print( pSum[e])
         }
         else{
             print(pSum[e]- pSum[s-1])
         }
```

Construct Psum[] For Given ar[N]

```
           0    1    2    3    4
ar[5] = {  3   -2    4    5    6 }
sum = 0    3    1    5    10   16
           ↓    ↓    ↓    ↓    ↓
psum[5]: { 3    1    5    10   16 }
```

Steps: Carry forward sum L → R

1. update sum
2. Storing in psum[]

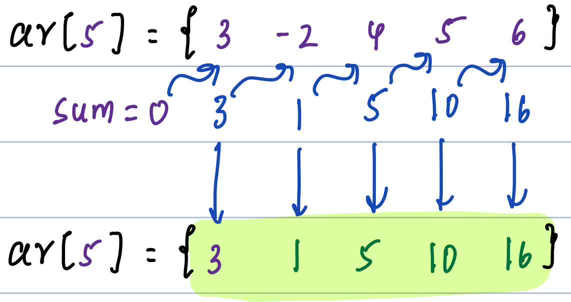
→ pfsum array.

TC: O(N+Q)   SC: O(N+5) ≈ O(N)

```
void RangeSum(int ar[], int N, int Qmat[][], int Q){
    long pSum[N]; // long = Storing sum values >> int range.
    long sum=0;
    for(int i=0;i<N;i++){ ⟶ TC: N
        sum=sum+ar[i]; //update
        pSum[i]=sum; //Storing
    }

    for(int i=0; i<Q; i++){ ⟶ TC: Q
        int s= Qmat[i][0], e=Qmat[i][1];
        if(s==0){ // sum: [0..e]
            print(pSum[e]);
        }
        else{
            print(pSum[e] - pSum[s-1]);
        }
    }
}
```

Note: Modifying same arr[] into Prefix arr[]



Issues in storing psum[] → arr[]

1. loose input arr[] information.
2. pfsum[] datatype & arr[] datatype might not match.

When can we Stre psum[] → arr[]

1. If arr[] is no longer needed.
2. if pfsum[] datatype & arr[] datatype is same.

Note: when multiples range queries, think in pfsum[]