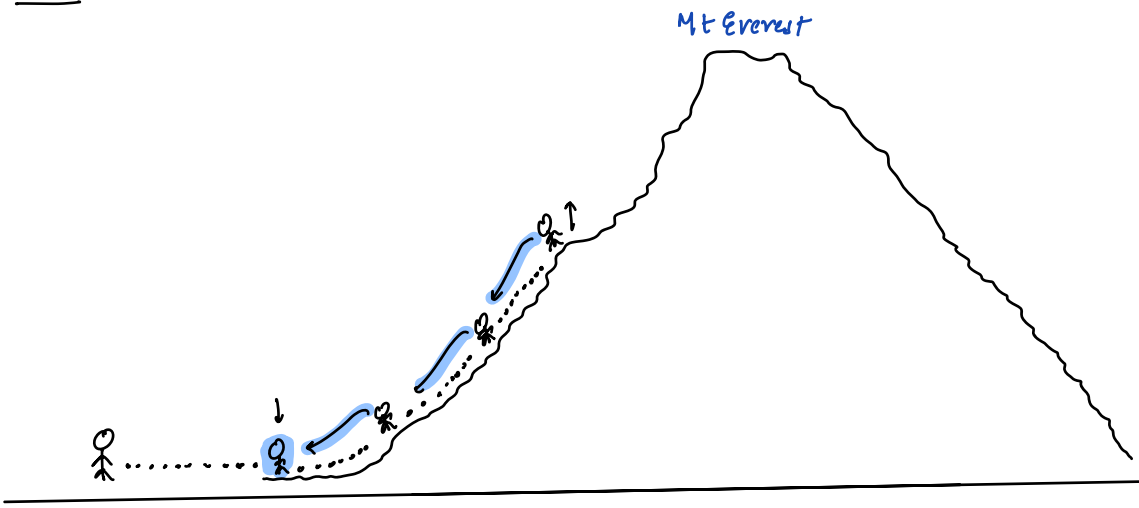
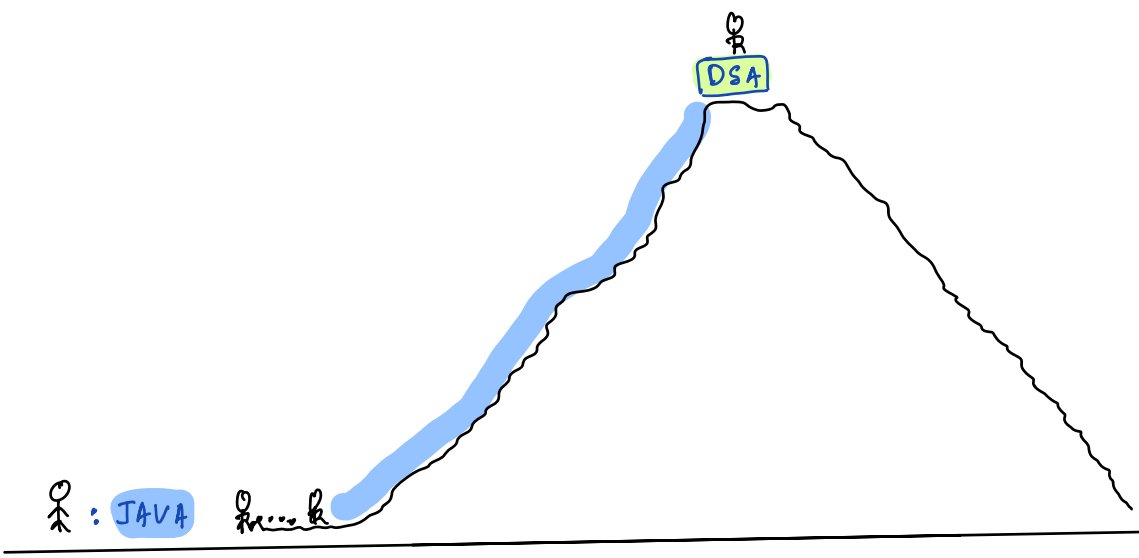
Story: Famous Mountain Climber: Edmund Hillary



He Said: Mountain you cannot grow as a person I can



Todays Content

a) Range Queries

b) Prefix Sum

c) Equilibrium Index

d) Even numbers in given range

Q1. Given ar[N] elements and mat[Q][2], each row in mat[][] is Query

i^{th} Query : mat[i][0] to mat[i][1] For every query Calculate

Sum of all index elements in ar[] from mat[i][0] ... mat[i][1]

Note: Store all query sums in ans[] & return ans[].

Constraints: $1 <= N, Q <= 10^5$

$-10^9 <= A[i] <= 10^9$

Note: At max sum of all ele: 10^5 ele, each ele = 10^9 = 10^{14} ⇒ long.

$0 <= mat[i][0] <= mat[i][1] < N$

Eg: ar[10]:

0	1	2	3	4	5	6	7	8	9
-3	6	2	4	5	2	8	-9	3	1

Q = 6

mat[6][2)

Sum[6]

	s	e
	0	1
0	4	8
1	3	7
2	1	3
3	7	7
4	3	6
5	0	4

0 : 9

1 : 10

2 : 12

3 : -9

4 : 19

5 : 14

Idea: For every query iterate & calculate & store in ans[].

```
long[] RangeSum(int ar[], int mat[][]){
    int Q = mat.length; // no: of rows
    long ans[Q]; //rough syntax
    for(int i=0; i<Q; i++){ // iterate each Query
        // ith Query Inf: In ith row
        int s = mat[i][0], e = mat[i][1];
        long sum=0;
        for(int j=s; j<=e; j++){
            sum = sum+ar[j]
        }
        ans[i] = sum;
    }
    return ans;
}
```

TC: Q * {Each Query Worst Case: N} = Q*N

SC: O(1)

Constraints: $1 <= N <= 10^5$ $1 <= Q <= 10^5$

↳ $N*Q = 10^5 * 10^5 = 10^{10} > 10^8$ TLE

Q: Given Indian Cricket Team Score, 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
Total Score:	2	8	14	29	31	49	65	79	88	97

Total runs scored in 10^{th} over: score[10] - score[9]

97 - 88 : 9

Total runs scored in 7^{th} over: score[7] - score[6]

65 - 49 : 16

Total runs scored in 3^{th} - 6^{th} : score[6] - score[2]

49 - 8 : 41

Total runs scored in 6^{th} - 10^{th} : score[10] - score[5]

97 - 31 : 66

Total runs scored in 5^{th} - 8^{th} : score[8] - score[4]

79 - 29 : 50

Total runs scored i^{th} over to j^{th} over : scores[j] - scores[i-1]

Idea: In Cricket, we are given total score till that Over.

Also called cummulative Sum

Idea: Cummulative sum from start = PrefixSum

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

Psum[i] = Sum of all arr elements from index [0..i]

mat[6][2] ans using Psum[]

	0	1		
0	4	8	Psum[8] - Psum[3] = 18-9 = 9	9
1	3	7	Psum[7] - Psum[2] = 15-5 = 10	10
2	1	3	Psum[3] - Psum[0] = 9-(-3) = 12	12
3	7	7	Psum[7] - Psum[6] = 15-24 = -9	-9
4	3	6	Psum[6] - Psum[2] = 24-5 = 9	19
5	0	4	Psum[4] - Psum[-1] *	
			Sum of ele: [0..4] = Psum[4]	14

```
S:E = if(S==0){ // sum [0..E]
        val = psum[E]
      }
      else{
        val = Psum[E] - Psum[S-1]
      }
```

Psum[i] = Sum of all arr[] from index 0..i

10:pm

Construct pSum[] ? for Given ar[N]

	0	1	2	3	4
ar[5] = {	3	-2	4	5	6 }

Psum[5] = { 3 1 5 10 }

PSum[0] = ar[0] = 3

PSum[1] = ar[0] + ar[1] = 1

Psum[1] = Psum[0] + ar[1]

PSum[2] = ar[0] + ar[1] + ar[2] = 5

Psum[2] = Psum[1] + ar[2]

PSum[3] = ar[0] + ar[1] + ar[2] + ar[3]

Psum[3] = Psum[2] + ar[3]

Psum[i] = Psum[i-1] + ar[i]

Q) Given ar[N] Create Psum[]

```
Psum[0] = ar[0]
for(int i=1 ; i< N; i++){
    Psum[i] = Psum[i-1] + ar[i]  // i=0 ⇒ Psum[-1] Error
}
```

Dry Run:

	0	1	2	3	4
ar[5] = {	3	-2	4	5	6 }
Psum[5] = {	3	1	5	10	16 }

```
long[] RangeSum(int ar[], int mat[][]){
    int N = ar.length;
    long pSum[N]; -> // Extra Space
    Psum[0] = ar[0]
    for(int i=1; i<N; i++){ ------------> O(N)
        Psum[i] = Psum[i-1] + ar[i]
    }
    // Ans Queries using Psum[]
    int Q = mat.length;
    long ans[Q];
    for(int i=0; i<Q; i++){ ------------> Q*O(1)
        // ith Query pnf pn on ith row ith:
        int s = mat[i][0], e = mat[i][1]
        long val = 0;
        if(s==0){ // sum
            val = psum[e]
        }
        else{
            val = Psum[e] - Psum[s-1]
        }
        ans[i] = val;
    }
    return ans;
}
```

Total TC: O(N+Q)

Total SC: O(N)

0	1
s	e

10:20pm

When to Use Prefin Sum:

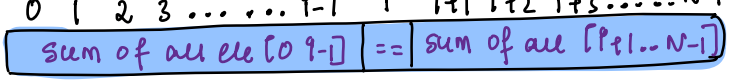
Whenever we see range sum queries, we can get PrefinSum.

Equibrium Inden : 50 mins

Given ar[N] elements, count no: of equilibrium inden

An inden i is said to be equilibrium inden if:

Sum of all elements on left of i^{th} inden = Sum of all elements on right of i^{th} inden



Note: if i == 0, leftsum = 0

if i == N-1, rightsum = 0

Constraints: $1 <= N <= 10^5$

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

Ex1: ar[4] = { 0: -3, 1: 2, 2: 4, 3: -1 } ans=1

	0	1	2	3
LeftSum =	0	-3	-1	3
Rig Sum =	5	3	-1	0

Ex2: ar[7] = { 0: -7, 1: 1, 2: 5, 3: 2, 4: -4, 5: 3, 6: 0 } ans=2

	0	1	2	3	4	5	6
LeftSum =	0	-7	-6	-1	1	-3	0
Rig Sum =	7	6	1	-1	3	0	0

Ideas: Construct Psum[] to solve problem

Psum[i] = sum of all elements from [0..i]

[s e] = Psum[e] - Psum[s-1]

[s = i+1, e = n-1] = Psum[n-1] - Psum[i+1-1]

Psum[i-1] == Psum[n-1] - Psum[i]

Idea: For every index, check if it's equilibrium or Not?

```
int equilibrium(int ar[]){
    int N = ar.length;
    long psum[N]; -> // Extra Space
    Psum[0] = ar[0]
    for(int i=1; i<N; i++){ ------> O(N)
        Psum[i] = Psum[i-1] + ar[i]
    }
    int c=0;
    for(int i=0; i<N; i++){ ------> O(N)*1
        // Check if ith index is Equilibrium
        Psum[i-1]  ==  Psum[N-1] - Psum[i]
        long left;
        if(i==0){ left=0}
        else{ left = Psum[i-1]; } // i=0 -> Psum[-1] Error
        long right = Psum[N-1] - Psum[i] // i=N-1 -> Psum[N-1] - Psum[N-1] = 0
        if(left == right){
            c=c+1
        }
    }
    return c;
}
```

TC: O(N+N) = O(2N)

SC: O(N+8) = O(N)

Q3. Given ar[N] elements and mat[Q][2], Each row in mat[][] is Query

i^{th} Query : mat[i][0] ... mat[i][1] For every query Calculate

No: of even elements in ar[] from index mat[i][0] ... mat[i][1]

Note: Store all query counts in ans[] & return ans[].

Constraints: $1 <= N, Q <= 10^5$

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

$0 <= L <= R < N$

0	1	2	3	4	5	6	7	8	9
2	4	3	7	9	8	6	3	4	9

Ex: ar[10] :

Q = 4

mat[4][2]

	0	1
0	4	8
1	3	9
2	2	7
3	0	4

ans[4]

0 3

1 3

2 2

3 2

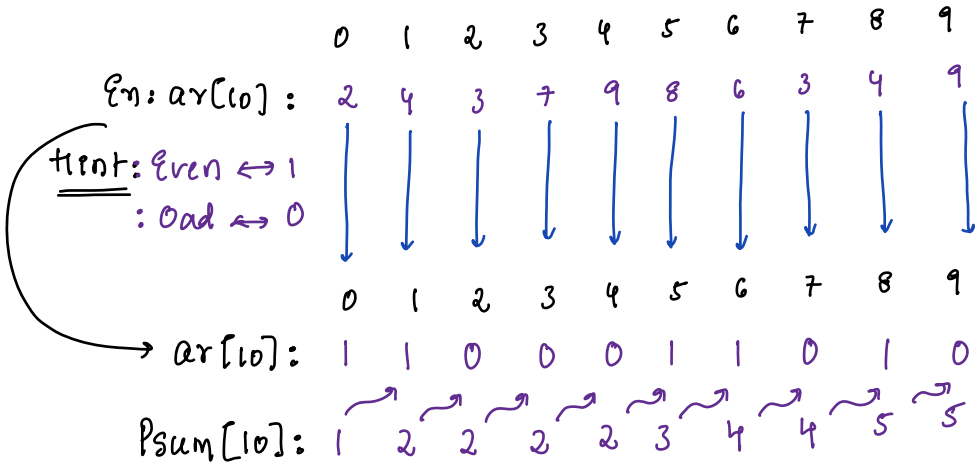
Idea: For each query iterate & get no:of even elements

```
int[] RangeSum(int ar[], int mat[][]){
    int Q = mat.length; // no: of rows
    int ans[Q]; //rough syntax
    for(int i=0; i<Q; i++){ // iterate each Query
        // ith Query Inf: In ith row
        int s = mat[i][0], e = mat[i][1];
        int c=0;
        for(int j=s; j<=e; j++){
            if(ar[j]%2==0){c=c+1}
        }
        ans[i] = c;
    }
    return ans;
}
```

TC: Q * { Worst Case we iterate N times } = Q*N

SC: O(1)

Constraints: $1 <= N <= 10^5$ $1 <= Q <= 10^5$ = $10^5 * 10^5 = 10^{10} > 10^8$



Psum[i] : Count of even numbers from [0...i]

mat[4][2]

Ans using Psum[]

	s=0	e=1	
0	4	8	= Psum[8] - Psum[3] = 3 — 3
1	3	9	= Psum[9] - Psum[2] = 3 — 3
2	2	7	= Psum[7] - Psum[1] = 2 — 2
3	0	4	= Psum[4] = 2 — 2

```
int[] RangeSum(int ar[], int mat[][]){        TC: O(2N+Q) = O(N+Q)
                                              SC: O(N)
    int N = ar.length;
    for(int i=0; i<N; i++){ ------------------> O(N)
        if(ar[i]%2==0){ ar[i]=1 }
        else{ ar[i]=0 }
    }

    int psum[N];
    Psum[0] = ar[0]
    for(int i=1; i<N; i++){ ------------------> O(N)
        Psum[i] = Psum[i-1] + ar[i]
    }

    // Ans Queries using Psum[]
    int Q = mat.length;
    int ans[Q];
    for(int i=0; i<Q; i++){ ------------------> Q*8 : O(Q)
        // ith Query pnf in on ith row ith :
        int s = mat[i][0], e = mat[i][1]
        int c=0
        if(s==0){ // sum
            c = psum[e]
        }
        else{
            c = Psum[e] - Psum[s-1]
        }
        ans[i] = c;
    }
    return ans;
}
```

0	1
s	e