Agenda

- Range queries
- Prefix sum
- Sum of even/odd indices
- Special index

QUIZ!

Find the Space Complexity [Big(O)] of the below program.

```
func(int N) { // 4 bytes
    int arr[10]; // 40 Bytes
    int x; // 4 bytes
    int y; // 4 bytes
    long z; // 8 bytes
    int arr[N]; // 4 * N bytes
}
```

Total add^n space : 56 + 4N

S.C : O(N)

QUIZ!

What does the following Pseudocode do ?

```
Function(arr[], N){
    i=0, j=N-1;
    while(i < j) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        i++;
        j--;
    }
}
```

Reversing the array.

QUIZ!

What does this algorithm do ?

```
Function (arr[], N, k){
    reverse(arr, N, 0, N-1);
    reverse(arr, N, 0, K-1);
    reverse(arr, N, K, N-1);
}
```

Rotate the array K times.

< Question > : Given an array of N integers and Q queries. For each query calculate the sum of elements in the range - [L , R]

Note : L and R are indices such that L ≤ R.

$1 \le N, Q \le 10^5$

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

arr[10] =

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

Queries

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

Brute Force Approach:

Q - size of Queries array

N - size of arr

```
void QuerySum( int Queries[][], int arr[]){
    for(i=0; i < Queries.length(); i++){
        l = Queries[i][0];
        r = Queries[i][1];
        sum = 0;
        for(j=l; j<=r; j++){
            sum += arr[j];
        }
        print(sum);
    }
}
```

i	j	# of Iter^n
0	[0 - N]	N
1	[0 - N]	N
2	[0 - N]	N
⋮	⋮	⋮
Q	[0 - N]	N

Total Itr^n = N+N--N

Q times
= N.Q

T.C : O(N.Q)

$10^5 \cdot 10^5 = 10^{10} > 10^8$ ↓ TLE

S.C : O(1)

- **Given Royal Challengers Bengaluru's cricket scores for first 10 overs of batting.**

OVERS	1	2	3	4	5	6	7	8	9	10
SCORE	2	8	14	29	31	49	65	79	88	97

QUIZ

- Runs scored in 7th over = Runs scored till 7th over - Runs scored in 6th over
  = 65 - 49 = 16
- Runs scored from 6 - 10th over = Runs scored till 10th over - Runs scored till 5th over
  = 97 - 31 = 66
- Runs scored in 10th over = score[10] - score[9] = 97 - 88 = 9
- Runs scored from 3 - 6th over = score[6] - score[2]
  = 49 - 8 = 41
- Runs scored from 4 - 9th over = score[9] - score[3]
  = 88 - 14 = 74
- Runs scored in lth - rth over = score[r] - score[l - 1]

Observations:

Here in case of cricket scoreboard, we are having cumulative sum / prefix sum b'coz of which queries can be answered in const time.

How to create psum()

psum[i] : Sum till i^{th} index

OR

Sum of elements from 0^{th} till i^{th} index.

QUIZ:

	0	1	2	3	4
arr[] =	2	5	-1	7	1
psum[] =	2	7	6	13	14

	0	1	2	3	4	5
arr[] =	10	32	6	12	20	1
psum[] =	10	42	48	60	80	81

BruteForce Approach:

```
psum[N];
for (i=0; i<N; i++){
    int sum = 0;

    for (j=0; j<=i; j++){
        sum += arr[j];
    }
    psum[i] = sum;
}
```

T.C : $O(N^2)$

S.C : $O(1)$

Observations:

psum[0] = arr[0]

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

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

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

psum[i] = psum[i-1] + arr[i] $\forall$ i > 0

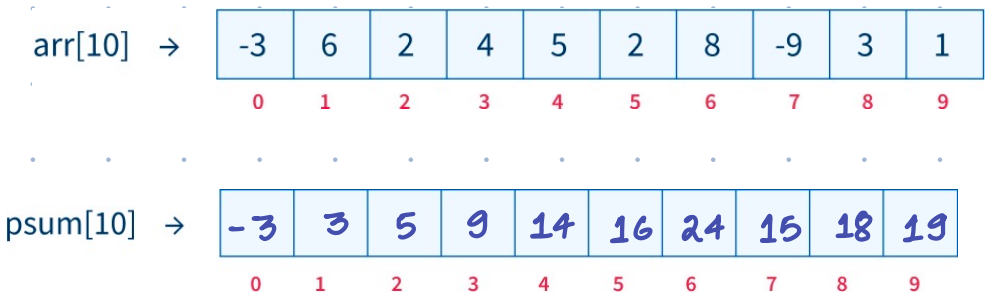
Optimised Code:

```
psum[N];
psum[0] = arr[0];

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

T.C: O(N)

S.C: O(1)

How to answer the queries ?



Queries - 5

L	R
4	8
3	7
1	3
0	4
7	7

→ psum[8] - psum[3] = 18 - 9 = 9

→ psum[7] - psum[2] = 15 - 5 = 10

→ psum[3] - psum[0] = 9 - (-3) = 12

→ psum[4] = 14

→ psum[7] - psum[6] = 15 - 24 = -9

Generalised eqⁿ to find Prefix Sum:

sum[L, R] = psum[R] - psum[L-1] L != 0

If (L == 0) sum[L, R] = psum[R]

```
void QuerySum( int Queries[][], int arr[]){

    // Create psum[]
       long psum[N];
      psum[0] = arr[0];

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

     // Answer Queries.
     for(i=0; i< Queries.length; i++){      → Q iterations
         l = Queries[i][0];
         r = Queries[i][1];

         if(l==0)
             sum = psum[r];
         else
             sum = psum[r] - psum[l-1];

         print(sum);
     }
}
```

T.C : O(N+Q)

S.C : O(N)

Range of Int : $[-2 \times 10^9, 2 \times 10^9]$

Range of Long : $[-9 \times 10^{18}, 9 \times 10^{18}]$

< *Question* > : Given an arr[N] and Q queries with start(s) and end(e) index. For every query print sum of all even indexed elements from s to e.

arr[ ] →

2	3	1	6	4	5
0	1	2	3	4	5

$1 \le N, Q \le 10^5$

$0 <= s, e <= N$

Queries

s	e	
1	3	→ 1
2	5	→ 5
0	4	→ 7
3	3	→ 0

Brute Force Approach :

```
for(i=0; i< Queries.length(); i++){
    l = Queries[i][0];
    r = Queries[i][1];
      sum=0;
    for(j=l; j<=r; j++){
        if(j%2==0){
            sum += arr[j];
        }
    }
    print(sum);
}
```

T.C : O(N * Q)

S.C : O(1)

Optimisation :

Range Queries → Prefix Sum

psum[i] : sum of even index elements from index 0 till i^{th} index.

arr[ ] →

0	1	2	3	4	5
2	3	1	6	4	5

psum[ ] =

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

psum[i] = psum[i-1] + arr[i] → i is even

psum[i] = psum[i-1] → i is odd

QUIZ:

arr[ ] =

0	1	2	3	4
2	4	3	1	5

psum[ ] =

0	1	2	3	4
2	2	5	5	10

Pseudocode :

```
// create psum[]
psum[N];
psum[0] = arr[0];
for(i=1 ; i<N ; i++) {
    if(i%2 == 0)
        psum[i] = psum[i-1] + arr[i];
    else
        psum[i] = psum[i-1];
}

// Answer the Queries
for(i=0 ; i< Queries.length ; i++) {
    s = Queries[i][0];
    e = Queries[i][1];

    if(s==0)
        sum = psum[e];
    else
        sum = psum[e] - psum[s-1];

    print(sum);
}
```

T.C : O(N+Q)

S.C : O(N)

< Question > : For all the queries, find the sum of odd indexed elements from s to e.

Special Index

< *Question* > : Given an arr[N], count the number of special indices in the array.

Special Index : Index after removing which,

Sum of even indexed elements = sum of odd indexed elements.

Constraints :

$1 \le N \le 10^5$

$-10^5 \le A[i] \le 10^5$

Sum of elements of arr $\le 10^9$

arr[ ] → [4 3 2 7 6 -2]
0 1 2 3 4 5

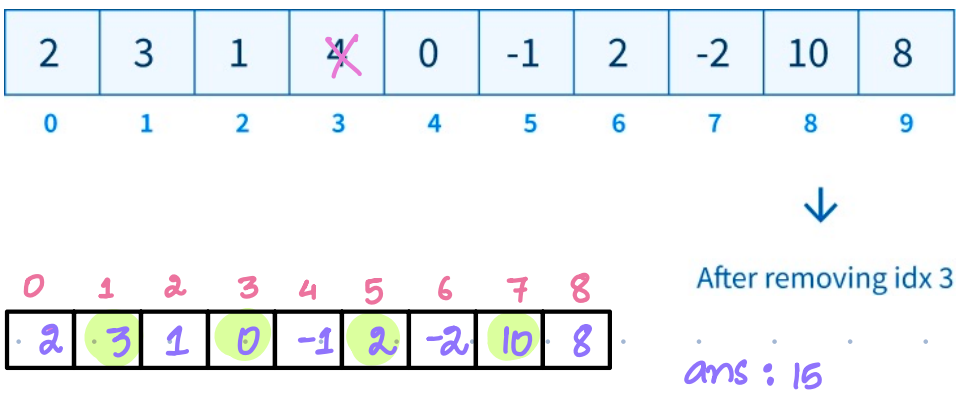
		S_e	S_o	
remove idx = 0 →	[4 3 2 7 6 -2] → [3 2 7 6 -2]	8	8	✓
remove idx = 1 →	[4 3 2 7 6 -2] → [4 2 7 6 -2]	9	8	✗
remove idx = 2 →	[4 3 2 7 6 -2] → [4 3 7 6 -2]	9	9	✓
remove idx = 3 →	[4 3 2 7 6 -2] → [4 3 2 6 -2]	4	9	✗
remove idx = 4 →	[4 3 2 7 6 -2] → [4 3 2 7 -2]	4	10	✗
remove idx = 5 →	[4 3 2 7 6 -2] → [4 3 2 7 6]	12	10	✗

ans : 2

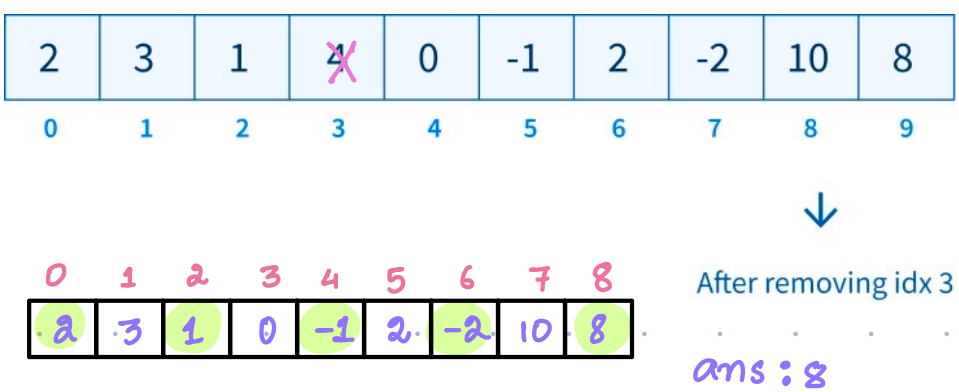
Quiz

1. What will be the sum of elements at ODD indices in the resulting array after removal of index 2 ?

2. What will be the sum of elements at ODD indices in the resulting array after removal of index 3 ?



3. What will be the sum of elements at EVEN indices in the resulting array after removal of index 3 ?



[BF Idea] -

For every index "i", remove element at i^{th} index & create new array from Remaining elements.

Calculate sum of even index elements (S_e) & sum of odd index elements (S_o)

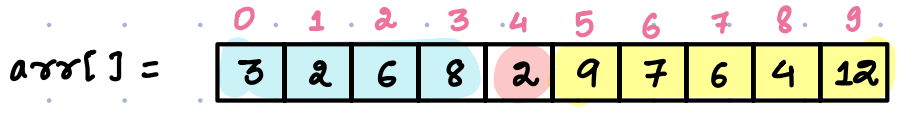
If $S_e == S_o$ → Increment count.

```
count = 0;
for (i = 0 ⟶ N) {
    // Create new array after removing element at ith index. (Cp[])

    for (j = 0 ⟶ N-1) {
        if (j % 2 == 0)
            Se += Cp[j];
        else
            So += Cp[j];
    }

    if (Se == So)
        count++;
}
return count;
```

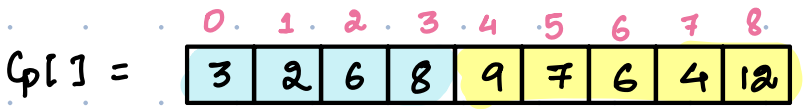
T.C : $O(N^2)$

S.C : $O(N)$

[Optimised Approach]



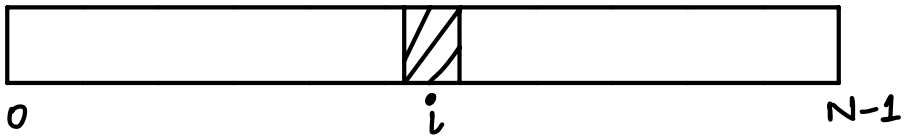
Even → Odd
Odd → Even



Sum of EVEN index elements after removing 4th index = Sum of EVEN index elements from index 0-3 in original array. + Sum of ODD index elements from index 5-9 in original array.

Sum of ODD index elements after removing 4th index = Sum of EVEN index elements from index 0-3 in original array. + Sum of ODD index elements from index 5-9 in original array.

Generalising:



After Removing i^{th} index:

$S_e = sum_{even\ indices}[0 \text{ --- } i-1] + sum_{odd\ indices}[i+1 \text{ ---- } N-1]$

$S_o = sum_{odd\ indices}[0 \text{ --- } i-1] + sum_{even\ indices}[i+1 \text{ --- } N-1]$

We can Precompute $Psum_{even}[\,]$ & $Psum_{odd}[\,]$.

$Psum_{even}[i]$: Sum of EVEN index elements from index 0 till ith index

$Psum_{odd}[i]$: Sum of ODD index elements from index 0 till ith index

$S_e = sum_{even\ indices}[0 \text{ --- } i-1] + sum_{odd\ indices}[i+1 \text{ ---- } N-1]$

$= \underbrace{Psum_{even}[i-1]}$ $+$ $\underbrace{Psum_{odd}[N-1] - Psum_{odd}[i]}$

$S_o = sum_{odd\ indices}[0 \text{ --- } i-1] + sum_{even\ indices}[i+1 \text{ --- } N-1]$

$= \underbrace{Psum_{odd}[i-1]}$ $+$ $\underbrace{Psum_{even}[N-1] - Psum_{even}[i]}$

Valid only for $i > 0$

If $i == 0$ (No elements before i^{th} index)

S_e = $Sum_{odd\ indices}^{[i+1 \text{ ---- } N-1]}$

$Psum_{odd}[N-1] - Psum_{odd}[0]$

S_o = $Sum_{even\ indices}^{[i+1 \text{ ---- } N-1]}$

$Psum_{even}[N-1] - Psum_{even}[0]$

```
int SpecialIndex(int[] arr, int n){
    // Create Psum_even[] & Psum_odd[]

    // Check for Special Index
    int count = 0;
    for(i=0; i<n; i++){
        if(i==0){
            if(Psum_odd[N-1] - Psum_odd[0] == Psum_even[N-1] - Psum_even[0])
                count++;
        }
        else{
            Se = Psum_even[i-1] + Psum_odd[N-1] - Psum_odd[i];
            So = Psum_odd[i-1] + Psum_even[N-1] - Psum_even[i];
            if(Se == So)
                count++;
        }
    }
    return count;
}
```

T.C : O(N)
S.C : O(N)