SCALER

Array - prefix sum

TABLE OF CONTENTS

Hello Everyone

Q.) Given an array with daily profit/loss from a particular stock. Calculate the total profit/loss over a given range of days.
The function should be able to handle multiple queries for different ranges.

	0	1	2	3	4	5	6
A = [	-5	10	20	40	50	-10	8]

9

Star day	End day	Profit / loss
0	6	113
1	4	120
4	5	40
0	0	-5

< *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. $1 \le N, Q \le 10^5$

	0	1	2	3	4	5	6	7	8	9
Arr[10] =	-3	6	2	4	5	2	8	-9	3	1

Q = 5

L	R	sum
4	8] → 0	9
3	7] → 1	10
1	2] → 2	8
0	4] → 3	14
7	7] → 4	-9

Bruteforce App: for every query iterate from L to R and calculate sum.

	0 L	1 R
0	3	7
1	0	4
2	1	2
3	7	7

row no.

L → Q[i][0]

R → Q[i][1]

```
function querySum(int[] A, int N,
                  int queries[][])
{
    // Q = Query.size
    for( i=0; i<Q ; i++)
    {
        int L = queries[i][0];
        int R = queries[i][1];
        int sum=0
        for(j=L ; j<=R; j++)
        {
            sum = sum + A[j];
        }
        print(sum);
    }
}
```

inner loop

L	R	
0	N-1	→ O(N)

1 Query → O(N)

TC: O(Q*N)

SC: 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

Q.1) Total run scored in 7^{th} over

= Run scored till 7^{th} over - Run scored till 6^{th} over

= S[7] - S[6] → over

= 65 - 49

= 16

Q2) Total run scored from 6^{th} - 10^{th} over.

= Run till 10^{th} over - Run till 5^{th} over

= 97 - 31 = S[10] - S[5].

= 66

Q 3) Total scored after 10^{th} over.

$= S[10] - S[9]$

$= 97 - 88$

$= 9$

Q.4) Total score from 3^{rd} to 6^{th} over.

$= S[6] - S[2]$

$= 49 - 8$

$= 41$

Q.5) Total score from 4^{th} to 9^{th} over.

$= S[9] - S[3]$

$= 88 - 14$

$= 74$

Total run scored from L - R over

$= S[R] - S[L-1]$

Score → Cumulative sum from starting of match.

Obs: Can we create Cricket like scoreboard for our integer array?

This Cricket like scoreboard is called as prefix sum or cumulative sum.

How we can create prefix sum array?

1) A[N] → PF[N]
2) PF[i] → sum of all element from starting till i^{th} index.

	0	1	2	3	4
A[5] = [	2	5	-1	7	1]
PF[5] =	2	7	6	13	14

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

// Given A[] and size N.
int PF[N].

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

	0	1	2	3	4	5	6	7	8	9
Gr[10] =	-3	6	2	4	5	2	8	-9	3	1
PF[10] =	-3	3	5	9	14	16	24	15	18	19

	L - R	Sum	PF[R] - PF[L-1]
0	4 8 :	PF[8] - PF[4-1] = 18 - 9 = 9	
1	3 7 :	PF[7] - PF[3-1] = 15 - 5 = 10	
2	1 3 :	PF[3] - PF[1-1] = 9 - (-3) = 12	
3	0 4 :	PF[4]	
4	7 7 :	PF[7] - PF[7-1] = 15 - 24 = -9	

if (L == 0) → PF[R].

① Sum[L R] = PF[R] - PF[L-1]

② Sum[0 R] = PF[R].

```
function querySum(int[] A, int N,
  int[][] queries)
{
    // Q -> query size.

// To Do -> Create PF[] -> O(N)

  for(i=0; i<Q; i++) -> O(Q)
  {
     int L = Queries[i][0];
     int R = Queries[i][1];

     if(L==0)
        sum = PF[R];
     else
        sum = PF[R] - PF[L-1];

     Print(sum);
  }
}
```

TC: $O(N+Q)$

SC: $O(N)$

BF	Optimised
TC: $O(N*Q)$ SC: $O(1)$	TC: $O(N+Q)$ SC: $O(N$

$N = 10^2$

$Q = 10^2$

$BF = 10^2 * 10^2$

$= 10^4$

optimise

$= 10^2 + 10^2$

$= 200$

Can we further optimize SC?

Yes, we can modify input array and store PF → SC $O(1)$

Break : 10.30 pm

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

	0	1	2	3	4	5
A[] =	2	3	1	6	4	5

Queries

L	R	
1	3	= A[2] = 1
2	5	= A[2] + A[4] = 1 + 4 = 5
0	4	= A[0] + A[2] + A[4] = 2 + 1 + 4 = 7
3	3	= 0

Approach1 : For every query iterate from L to R, if index is even, add it's element to the sum.

Tc: O(Q*N)

SC: O(1)

Approach 2: Create PFsum[] → sum of even index element.

① Range sum
② Multiple queries.

↓

Prefix sum

	0	1	2	3	4	5
A[6] =	2	3	1	6	4	5
PF[6] =	2	2	3	3	7	7

PF[i] = sum of all even index elements from 0 to i^{th} index.

L	R	PF[R] − PF[L−1]
1	3	PF[3] − PF[1−1] = 3−2 = 1
2	5	PF[5] − PF[2−1] = 7−2 = 5
0	4	PF[4] = 7
3	3	PF[3] − PF[3−1] = 3−3 = 0

```
void sumofEvenIndex(int[] A, int N,
  int Q, int[][] queries)
{
    int PF[N]          <-- SC: O(N)
    PF[0] = A[0]

    for(i=1; i<N; i++)
    {
        if(i%2 == 0)
        {
            PF[i] = PF[i-1] + A[i];
        }
        else
        {
            PF[i] = PF[i-1];
        }
    }
```

TC: O(N)

```
for (i=0; i<Q; i++)
{
    int L = queries[i][0];
    int R = queries[i][1];

    if (L==0)
        sum = PF[R];
    else
        sum = PF[R] - PF[L-1];

    print(sum)
}
```

TC: O(Q)

Q

L	R
0	3
4	5
1	3
1	2

TC: O(N+Q)

SC: O(N)

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.

Count = ~~2~~ 2

remove idx = 0 → [4 3 2 7 6 -2] → [3 2 7 6 -2] SE 8 SO 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 ✗

Quiz

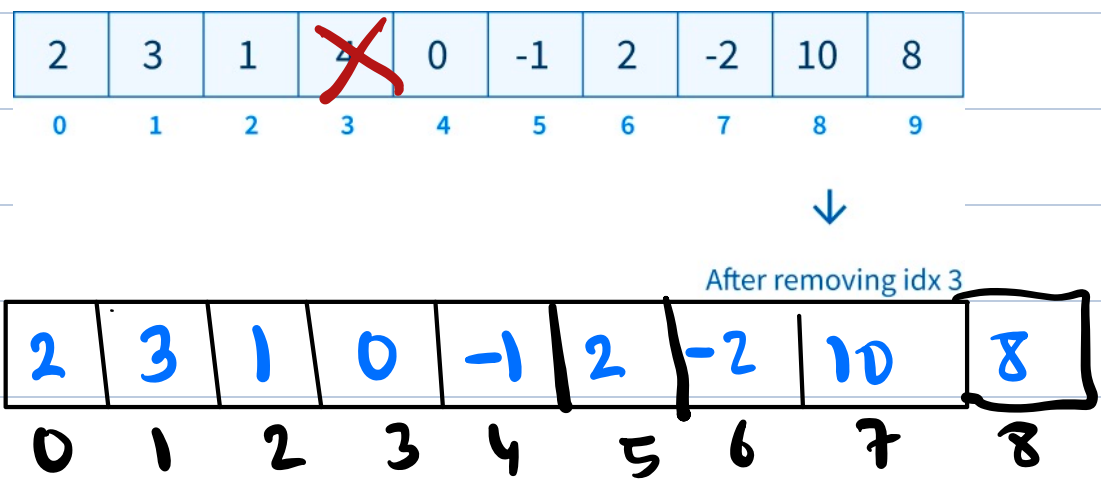
1.

$= A[1] + A[3]$

$= 1 + 10$

$= 11$

2.



Odd sum = 15

Even sum = 8

[BF Idea] - Go to every index and check whether it is special index.

① Delete element → shifting all right ele.

O(N)

② Iterate to calculate SE and SO.

O(N)

TC: $O(N^2)$

SC: O(1)

Optimized approach: Faster approach to check whether index is special or not.

We can't delete element, we want to find a way without deleting.

	0	1	2	3	4	5	6	7	8	9
A →	2	3	1	4	0	-1	2	-2	10	8

unchanges

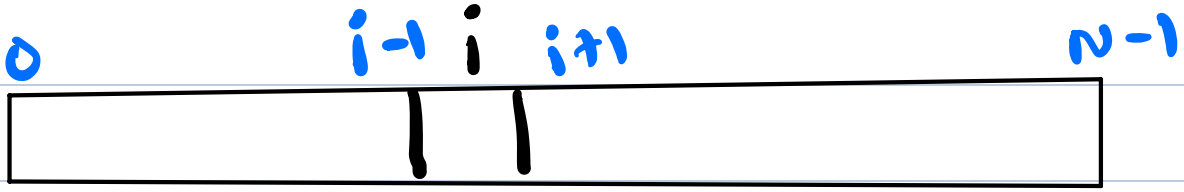
shifting by a place.

After removal

2	3	1	0	-1	2	-2	10	8
0	1	2	3	4	5	6	7	8

Even index → odd

odd index → Even



After removal of i index =

sum of Even index = sum of even index in range 0 to i-1 + sum of odd index in rang i+1 to n-1.

Sum of odd index = sum of odd index in rang 0 to i-1 + sum of even index in range i+1 to n-1.

TODO PFE[] → O(N)
TODO PFO[] → O(N)

```
int count = 0

for (i=0 ; i<N; i++)        → O(N)
{
    //ith index

int evenSum = getSum(PFE, 0, i-1)
            + getSum(PFO, i+1, n-1)

int oddSum = getsum(PFO, 0, i-1)+
             getsum(PFE, i+1, n-1);
```

```
if(evenSum == oddSum)
{
    count++;
}
}return count;
```

```
int getSum(int[] PF, int L, int R)
{
    if (R<0) return 0;
    if(L==0)
        return PF[R];
    else
        return PF[R]-PF[L-1];
}
```

TC: O(N)
SC: O(N)

	0	1	2	3	4	5	6	7	8	9
A →	2	3	1	4	0	-1	2	-2	10	8
PFE →	2	2	3	3	3	3	5	5	15	15

	0	1	2	3	4	5	6	7	8	9
A →	2	3	1	4	0	-1	2	-2	10	8
PFO →	0	3	3	7	7	6	6	4	4	12

```
for (i=0 ; i<N; i++)                                   → O(N)
{
    //ith index

int evensum = getsum(PFE, 0, i-1)
            + getsum(PFO, i+1, n-1)

int oddsum = getsum(PFO, 0, i-1)+
             getsum(PFE, i+1, n-1);

if(evensum == oddsum)
{
    count++;
}
}return count;
```

```
int getsum(int[] PF, int L, int R)
{
    if (R<0) return 0;
    if(L==0)
        return PF[R];
    else
        return PF[R]-PF[L-1];
}
```

Array - prefix sum

i = 0 evensum = 0 + 12 = 12

oddsum = 0 + 13 = 13

12 == 13 (no)

i = 1 evensum = 2 + 9 = 11

oddsum = 0 + 13 = 13

11 == 13 (no)

i = 2 evensum = 2 + 9 = 11

oddsum = 3 + 12 = 15

11 == 15 (no)

i = 3 evensum = 3 + 5 = 8

oddsum = 3 + 12 = 15

8 == 15 (no)

i = 4 evensum = 3 + 5 = 8

oddsum = 7 + 12 = 19

Doubt session