Today's content

(i) Prefix sum

(ii) Queries in range

(iii) Special Index / Balanced Index / Fair Array

(a) Sum of even indices in the array.

(b) sum of odd indices in the array.

Q1) Given n array elements, return PF[] where

PF[i] = sum { arr[0], arr[1], arr[2] --- arr[i] } for all i.

ex:

	0	1	2	3	4
arr[5] =	[5	2	7	-3	8]
PF[5] =	[5	7	14	11	19]

ex:

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

Idea: For every pF[i], iterate from [0...i] and find the sum.

code:

```
long
~~int~~[]   prefixSum(int ar[N])
    long
    ~~int~~[]  pf[N];
    for(i=0; i<N; i++)
        long
        ~~int~~ sum=0
        for(j=0; j<=i; j++)
            sum = sum+ar[j]
        PF[i] = sum
    return pF
```

TC: $O(N^2)$

SC: $O(N)$.

Optimization

ar[N] ⟶ pF[N].

	0	1	2	3	4
arr[5] =	5	2	7	-3	8
pF[5] =	5	7	14	11	19

1. pF[0] = ar[0]
2. pF[1] = ar[0] + ar[1]

   pF[1] = pF[0] + ar[1]
3. pF[2] = ar[0] + ar[1] + ar[2]

   pF[2] = pF[1] + ar[2]
4. pF[3] = ar[0] + ar[1] + ar[2] + ar[3]

   pF[3] = pF[2] + ar[3]

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

code:

```
int[]    prefixSum( int ar[N])
        int[]  pf [N];
        pF[0] = ar[0];
        for(i=1; i<N; i++)
            pF[i] = pF[i-1] + ar[i]
        return pF
```

TC: O(N)

SC: O(N)

Modify given array

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

TC: O(n)
SC: O(1)

code:

```
int[] prefixSum(int ar[N])
    for(i=1; i<N; i++)
        ar[i] = ar[i-1]+ar[i]
    return ar
```

Disadvantages

1. You're losing the original array.
2. Datatype ar → int[]
   pF[i] → This may overflow.

20: Given N array elements and Q queries.

For each query, calculate sum of all the elements in the given range.

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

Q = 6.

L	R	Ans
4	8	9
3	7	10
1	3	12
0	4	14
6	9	3
7	7	-9

idea:

```
void rangeSum( int ar[N] , int Q, int L[Q], int r[Q])
    for (i=0; i<Q; i++)
        int s = L[i] , int e = r[i]
        int sum=0
        for (j=s; j<=e; j++)
            sum = sum + ar[j]
        print(sum)
```

TC: O(Q*N)

SC: O(1)

Optimize?

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

sum[3 - 7] = sum[0-7] - sum[0-2]

sum[3-7] = pF[7] - pF[2].

sum [6 -9] = sum[0-9] - sum[0-5]

sum[6-9] = pF[9] - pF[5]

sum(s-e) = pF[e] - pF[s-1]

If s=0,

sum[0 - 7] = pF[7] - pF[-1] ✗

sum[0-7] = pF[7] ✓

s==e ✓✓

$$\text{sum}[s\text{-}e] = \begin{cases} s=0; & pF[e] \\ s!=0; & pF[e]-pF[s-1] \end{cases}$$

"REMEMBER THIS".

```
void   rangeSum (int ar[N], int Q, int L[Q], int r[Q])
       int[] pF[N] // TO-DO. populate it; O(N)

       for(i=0; i<Q; i++)
            int s= L[i], int e= r[i]
            if(s= =0)
                print(pF[e])
            else
                print( pF[e]-pF[s-1])
```

TC: O(N+Q)

SC: O(N)

Q3. Given an array ar[N] and Q queries, for each query [L-R], get the sum of all even index elements in given range.

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

index % 2 == 0 => even
else odd index.

Q = 4

L	R	Sum.
1	4	-2
2	7	6
3	8	14
0	4	1

idea: For every query, iterate from [L[i] -- R[i]] and sum up the even index elements only.

```
void    evenSum( int ar[N] , int Q, int L[Q], int r[Q])
        for (i=0; i<Q; i++)
            int s= L[i] , int e= r[i]
            int sum=0
            for (j=s; j<=e; j++)
                if(j % 2==0)
                    sum= sum+ar[j]
            print (sum)
```

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

idea2: Make all odd indices as zero.

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

odd indices value = 0.

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

	0	1	2	3	4	5	6	7	8
pFEven[9]:	3	3	4	4	1	1	9	9	18

pFEven[4] ⟶ sum of even index ele's from [0-4].

pFEven[7] ⟶ sum of even index ele's from [0-7].

pFEven[i] ⟶ sum of even index ele's from [0-i].

sum[1-4] (even index) = sum(0-4)(even index) - sum(0-0)(even index).

sum(1-4). = pFEven[4] - pFEven[0].

Q = 4

L	R	Sum.
1	4	pFEven[4]-pFEven[0]
2	7	pFEven[7]-pFEven[1]
3	8	pFEven[8]-pFEven[2]
0	4	pFEven[4]

Code:

```
void  evenSum( int ar[N] , int Q, int L[Q], int r[Q])

    // create pFEven.
    int[] pFEven[N];
    pFEven[0] = ar[0];
    for (i=1; i<N; i++)
        if(i%2==0)
            pFEven[i] = pFEven[i-1] + ar[i]
        else
            pFEven[i] = pFEven[i-1]

    for (i=0; i<Q; i++)
        int s = L[i], e = r[i]
        if (s==0)
            print(pFEven[e])
        else
            print(pFEven(e) - pFEven(s-1))
```

Populating pFEven.

O(N)

For Q iterations,
reading sum(s-e) for even
indices.

O(Q)

TC: O(N+Q)

SC: O(N)

Q4. Given an array ar[N] and Q queries, for each query [L, R] and get sum of all odd index elements.

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

index % 2 == 0 => even
else odd index.

Q = 4

L	R	Sum.
1	4	8
2	7	12
3	8	12
0	4	8

$$\text{Sum}[s\text{-}e] \begin{cases} s == 0, & \text{pFOdd}(e) \\ s\,!= 0, & \text{pFOdd}(e) - \text{pFOdd}(s-1) \end{cases}$$

TO-DO

Special Index.

An index is said to be special index, if after deleting that index

sum of all even index = sum of all odd index.

o/p: Count no. of special indices.

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ \text{Ex: } ar[6] : & [4 & 3 & 2 & 7 & 6 & -2] \end{array}$$

delete index 0.

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [3 & 2 & 7 & 6 & -2] \end{array}$$

$S_e = 8$, $S_o = 8$, $c = 1$.

delete index 1

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [4 & 2 & 7 & 6 & -2] \end{array}$$

$S_e = 9$, $S_o = 8$

delete index 2

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [4 & 3 & 7 & 6 & -2] \end{array}$$

$S_e = 9$, $S_o = 9$, $c = 2$.

delete index 3

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [4 & 3 & 2 & 6 & -2] \end{array}$$

$S_e = 4$, $S_o = 9$

delete index 4

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [4 & 3 & 2 & 7 & -2] \end{array}$$

$S_e = 4$, $S_o = 10$.

delete index 5

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & 4 \\ ar[5] : & [4 & 3 & 2 & 7 & 6] \end{array}$$

$S_e = 12$, $S_o = 10$.

ans: 2.

idea1: For every index i, create Cp[N-1] [ar[i] is removed]
Calculate Se and So, if (Se == So), C++.
return count.

Code:

```
int specialIndex(int ar[N])
    int c=0
    for(i=0; i<N; i++)
        // create Cp[N-1]
        // copy all elements except ar[i].
            ar[N] : 0, 1, 2 - - - i, -- N-1
            Cp[N-1] = [                    ] ∈ N-1
        int Se=0, So=0.
        // iterate over Cp array, find Se & So.
        if (Se == So)
            C++
    return c
```

TC: O(N*2N)

TC: O(N^2)

SC: O(N)

Ex1:

ar[10] :

0	1	2	3	4	5	6	7	8	9
3	2	6	8	2	9	7	6	4	12

even → odd
odd → even.

Delete index 4.

cp[9] :

0	1	2	3	4	5	6	7	8
3	2	6	8	9	7	6	4	12

$S_e \rightarrow \quad Cp\, S_e[0-8] = Cp\, S_e[0-3] + Cp\, S_e[4-8]$

$$\boxed{S_e = ar\, S_e[0-3] + ar\, S_o[5-9]}$$

$S_o \rightarrow \quad Cp\, S_o[0-8] = Cp\, S_o[0-3] + Cp\, S_o[4-8]$

$$\boxed{S_o = ar\, S_o[0-3] + ar\, S_e[5-9]}$$

Ex2:

ar[12] :

0	1	2	3	4	5	6	7	8	9	10	11
2	1	3	0	6	7	3	4	5	6	10	2

Delete 5^{th} index.

Cp[11] :

0	1	2	3	4	5	6	7	8	9	10
2	1	3	0	6	3	4	5	6	10	2

$S_e = Cp\, S_e[0-10] = Cp\, S_e[0-4] + Cp\, S_e[5-10]$

$$\boxed{S_e = ar\, S_e[0-4] + ar\, S_o[6-11]}$$

$S_o = Cp\, S_o[0-10] = Cp\, S_o[0-4] + Cp\, S_o[5-10]$

$$\boxed{S_o = ar\, S_o[0-4] + ar\, S_e[6-11]}$$

6) Generalization.

ar[N] :

0	1	2	3	i-1	i	i+1	...	n-1
a_0	a_1	a_2	a_3	a_{i-1}	a_i	a_{i+1}	a_{i+2} ---	a_{n-1}

Delete i^{th} index.

cp[N-1] :

0	1	2	3	i-1	i	...	n-2
a_0	a_1	a_2	a_3	a_{i-1}	a_{i+1}	a_{i+2} ---	a_{n-1}

$$S_e[0..N-1] = cp\,S_e[0..(i-1)] + cp\,S_e[i..(n-2)]$$

$$S_e = ar\,S_e[0..(i-1)] + ar\,S_o[(i+1)..(n-1)]$$

$$S_o[0..N-1] = cp\,S_o[0..(i-1)] + cp\,S_o[i...(n-2)]$$

$$S_o = ar\,S_o[0..(i-1)] + ar\,S_e[(i+1)..(n-1)]$$

$$S_e = ar\,S_e[0..(i-1)] + ar\,S_o[(i+1)..(n-1)]$$

$$= S_e[0 \quad i-1] + S_o[(i+1) \quad (n-1)]$$

$$S_e[\overset{s}{L} \; \overset{e}{R}] = \begin{cases} L==0, & pF_e(\overset{e}{R}) \\ L!=0. & pF_e(\overset{e}{R}) - pF_e(\overset{s-1}{L-1}) \end{cases}$$

$$S_e = pFEven[i-1] + pFOdd[n-1] - pFOdd[i]$$

$$S_o = ar\,S_o[0..(i-1)] + ar\,S_e[(i+1)..(n-1)]$$

$$= S_o[0 .. i-1] + S_e[(i+1)..(n-1)]$$

$$S_o = pFOdd[i-1] + pFEven[n-1] - pFEven[i]$$

Code:

```
int specialIndexCount(int ar[N])
    int pFE[N], pFO[N] // create, populate it.
    int c = 0
    for(i=0; i<N; i++)
        // we are deleting i^th index.
        int Se = pFOdd(n-1) - pFOdd(i)
        int So = pFEven(n-1) - pFEven(i)
        if(i!=0)
            Se = Se + pFEven(i-1)
            So = So + pFOdd(i-1)
        if(Se==So)
            c++
    return c
```

$S_e =$ pFEven[i-1] + pFOdd[n-1] - pFOdd[i]

$S_o =$ pFOdd[i-1] + pFEven[n-1] - pFEven[i]

TC : O(N)

SC : O(N)