Interesting Problems based on prefixSum

→ 10

→ 20 → {Google}

Q) Given N array elements & Q queries, for each query iterate & get 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

[% 2 == 0 ⇒ even
0 ⇒ even]

Q = 4

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

Ideal: For every query L R get sum of all even indices

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

```
// Given ar[N]
Read Q
for(i = 0; i < Q; i++){ ------> Q times
    Read L, R
    int sum=0
    for(int j= L; j <= R; j=j+1){
        if( j % 2 == 0){
            sum = sum + ar[j]
        }
    }
    print(sum)
}
```

→ O(N)

→ Revisim: { Using pf[] sum L-R }

L R → { if(L == 0) { print(pf[R]) }
else { print(pf[R] - pf[L-1] } }

Idea2: → { Make all odd indices zero }

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

↓ Step1 : → { Even without step, while constructing pfeven[] values }

ar[9] =	3	0	1	0	-3	0	8	0	9

↓ Step2:

pfeven[9] =	3	3	4	4	1	1	9	9	18

L R pfeven[R] - pfeven[L-1] : val

1 4 = pfeven[4] - pfeven[0] = -2

2 7 = pfeven[7] - pfeven[1] = 6

3 8 = pfeven[8] - pfeven[2] = 14

0 4 = pfeven[4] = 1

Pseudo Code :

1) Given ar[N]

Pfeven[N];

Pfeven[0] = ar[0]

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

→ TC: O(N)

Read Q

```
for(int i= 0 ; i < Q; i=i+1){
    Read L R
    if(L==0) { print( pfeven[R])}
    else{ print( pfeven[R] - Pfeven[L-1])}
}
```

→ TC: O(Q)

Overall TC: O(N+Q)

SC: O(N)

Trace:

	0	1	2	3	4	5	6
ar[7] =	3	2	6	1	2	1	3
Pfeven[7] =	3	3	9	9	11	11	14

→ A P J Abdul Kalam Sir:

↳ { Before you ask, try it out on your own }

Q) Given N array elements & Q queries, for each query iterate & get sum of all **odd 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

[%2==0 ⇒ even, 0 ⇒ even]

Q = 4

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

BF: For every query L R iterate & get odd index sum

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

Idea2:

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

Step1: replace even index = 0 } Without replacing we can get pfodd[]

ar[9] =	0	2	0	6	0	2	0	4	0

Pfodd

	0	1	2	3	4	5	6	7	8
pfodd[9] =	0	2	2	8	8	10	10	14	14

L	R	Pfodd[R] - Pfodd[L-1]
1	4	= Pfodd[4] - Pfodd[0] = 8
2	7	= Pfodd[7] - Pfodd[1] = 12
3	8	= Pfodd[8] - Pfodd[2] = 12
0	4	= Pfodd[4] = 8

Given ar[N]

Pfodd[N];

Pfodd[0] = 0

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

Read Q

```
for(int i = 0; i < Q; i = i+1){
    Read L R
    if(L==0){ print(Pfodd[R])}
    else{ print(Pfodd[R] - Pfodd[L-1])}
}
```

TC: O(N+Q)

SC: O(N)

Trace:

	0	1	2	3	4	5	6
ar[7] =	3	2	6	1	2	1	3
Pfodd[7] =	0	2	2	3	3	12	12

→

1. Pfeven[ ]
2. Pfodd[ ]
3. Queries using Pfeven[] or Pfodd[]

→

Special Index: → {HARD} → {Google}

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

Sum of all even index = Sum of all odd index

Count no: of special index are there?

	0	1	2	3	4	5
Eg1: ar[6] =	4	3	2	7	6	-2

delete index 0

	0	1	2	3	4
cp[5] =	3	2	7	6	-2

$S_e = 8 \quad S_o = 8, \quad C = C + 1$

delete index 1

	0	1	2	3	4
cp[5] =	4	2	7	6	-2

$S_e = 9, \quad S_o = 8$

delete index 2

	0	1	2	3	4
cp[5] =	4	3	7	6	-2

$S_e = 9, \quad S_o = 9 \quad C = C + 1$

delete index 3

	0	1	2	3	4
cp[5] =	4	3	2	6	-2

$S_e = 4 \quad S_o = 9$

delete index 4

	0	1	2	3	4
cp[5] =	4	3	2	7	-2

$S_e = 4, \quad S_o = 10$

delete index 5

	0	1	2	3	4
cp[5] =	4	3	2	7	6

$S_e = 12 \quad S_o = 10$

Idea: for index i, create cp[N-1], where ar[i] is removed calculate S_o & S_e & compare them.

Pseudocode:

```
int specialIndex(int ar[]) {
    int n = ar.length
    int c = 0
    for (int i = 0; i < n; i = i+1) {        → N times
        // Create new array cp[N-1] where i^th index of original ar[] is not present
        ar[N]: 0 1 2 ... i-1 | i | i+1 i+2 ... N-1
        cp[N-1]:
        int Se = 0, So = 0
        // Iterate on cp[] & get Se So          → O(N)
        // if (Se == So) { c = c+1 }            → O(1)
    }
    return c;
}
```

TODO!

Doubts Session

O(N)

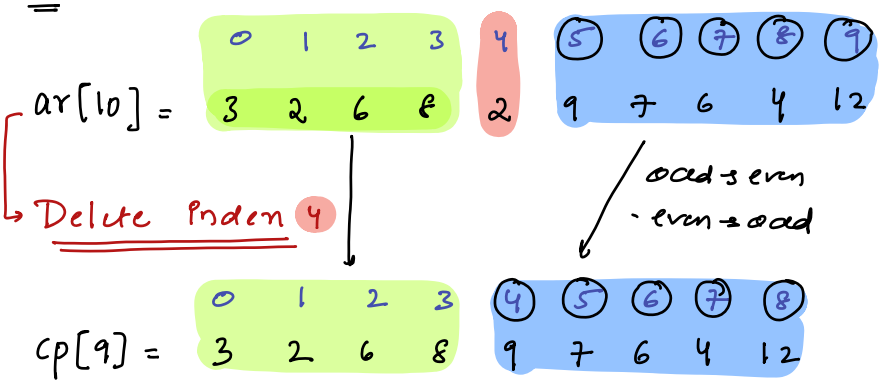
TC: $N * \{ N + N+1 \} \Rightarrow O(N^2)$

deleting

Se & So

SC: O(N)

// Idea:



$S_e = cp\,Se[0\ 8] = cp\,Se[0\ 3] + cp\,Se[4\ 8]$

36 9 27

After deleting 4th Index

$Se = arSe[0\ 3] + arSo[5\ 9]$

9 27

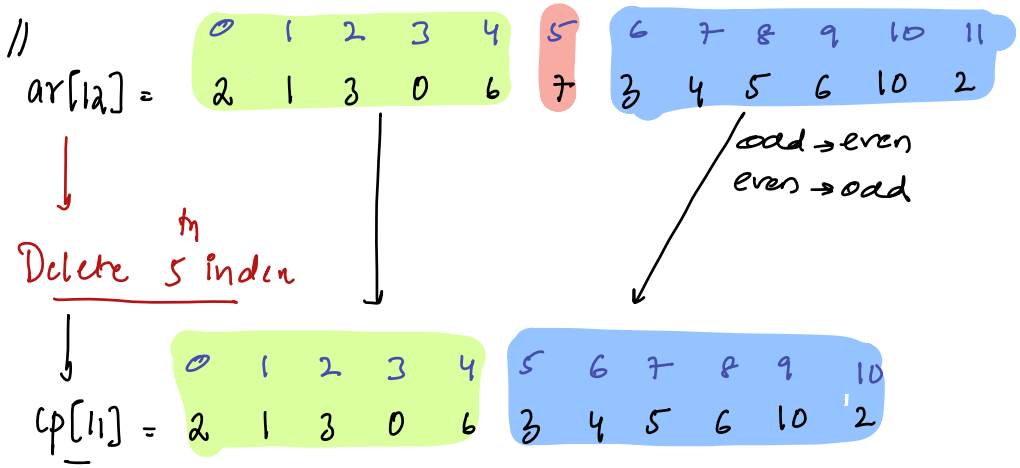
$S_o = cp\,So[0\ 8] = cp\,So[0\ 3] + cp\,So[4\ 8]$

10 11

After deleting 4th Index

$So = arSo[0\ 3] + arSe[5\ 9]$

10 11



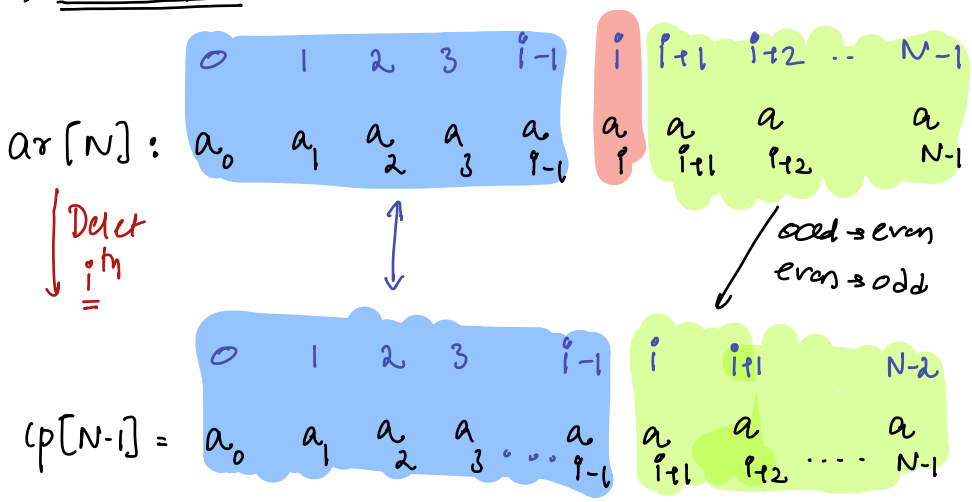
$Se = Cp\,Se[0\ 10] = Cp\,Se[0\ 4] + Cp\,Se[5\ 10]$

$Cp\,Se[0\ 4] \rightarrow 11 \rightarrow 11$

$Cp\,Se[5\ 10] \rightarrow 12 \rightarrow 12$

$arSc[0\ 4] + arSo[6\ 11]$

// generalization



$S_e = cpS_e[0, N-2] = cpS_e[0, i-1] + cpS_c[i, N-2]$

Total S_e after Deleting i^{th} index $= arS_c[0, i-1] + arS_o[i+1, N-1]$

$S_o = cpS_o[0, N-2] = cpS_o[0, i-1] + cpS_o[i, N-2]$

Total S_o after Deleting i^{th} index $= arS_o[0, i-1] + arS_c[i+1, N-1]$

Final expressions: After deleting i^{th} index

↳ Step 1: Construct pfe[N] & pfo[N]

$$S_e = S_e[\underset{L}{0}\ \underset{R}{i-1}] + S_o[\underset{L}{i+1},\ \underset{R}{N-1}]$$

$$S_e = pfe[i-1] + pfo[N-1] - pfo[i]$$

$$S_o = S_o[0\ i-1] + S_e[i+1,\ N-1]$$

$$S_o = pfo[i-1] + pfe[N-1] - pfe[i]$$

Pseudocode:

```
int specialIndex(int ar[]){
    int n = ar.length
    int pfe[n], pfo[n]
    // Get values for pfe[] & pfo[] TODO
    int c = 0;
    for(int i = 0; i < N; i++){
        // We are deleting index i        pfe[i-1] +
        int Se = pfo[N-1] - pfo[i]
        if(i != 0){ Se = Se + pfe[i-1] }
        int So = pfe[N-1] - pfe[i]
        if(i != 0){ So = So + pfo[i-1] }
        if( Se == So ){ c = c+1 }
    }
    return c;
}
```

$S_e = pfe[i-1] + pfo[N-1] - pfo[i]$ ↳ it has edge case $i = 0$

$S_o = pfo[i-1] + pfe[N-1] + pfe[i]$ ↳ it has edge case $i = 0$

TC: $O(N + N + N) \Rightarrow O(N)$ SC: $O(N + N) \Rightarrow O(N)$

Edge Cases:

$i = 0$: array out of bounds

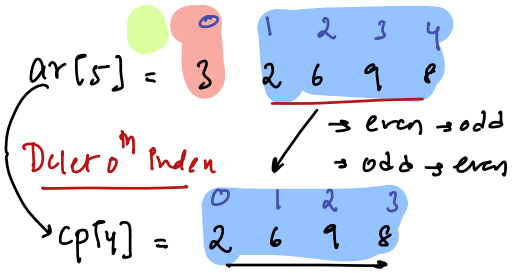
$S_e = pfe[i-1] + pfo[N-1] - pfo[i]$ ($pfe[i-1]$: $i-1 = 0$)

$S_o = pfo[i-1] + pfe[N-1] - pfe[i]$ ($pfo[i-1]$: $i-1 = 0$)

$i = N-1$: code works

$S_e = pfe[i-1] + pfo[N-1] - pfo[i]$

$S_o = pfo[i-1] + pfe[N-1] - pfe[i]$ ($pfo[i-1]$: $i-1 = 0$)



S_e = Pfe[-1] + Pfo[1-4]

= Pfe[0] + Pfo[1-4]

Doubts ? → Pick k elements

ar = [5, -2, 2, 3, 6, 1, 8] k = 4 [N-k N-1]

// Say N k =

C_1 = S[0 k-1] → done

C_1 = S[0 3]

C_2 = S[0 2] + S[6]

C_3 = S[0 1] + S[5 6]

C_4 = S[0 0] + S[4 6]

C_5 = S[3 6]

C_2 = S[0 k-2] + S[N-1 N-1]

C_3 = S[0, k-3] + S[N-2 N-1]

C_4 = S[0, k-4] + S[N-3 N-1]

⋮

C_{last} = S[0 0] + S[N-k+1 N-1]

C_{last} = S[N-k, N-1] → done