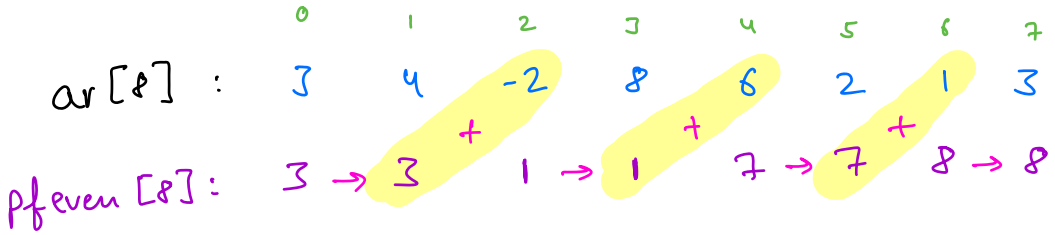
Arrays: Prefix Sum 2

No HW or Asgn.

- Queries on pfeven
- Special index

pfeven

Q4 Given N array elements & Q queries, for each query calculate sum of all even indices in given range.



Queries

[2 6]: $a_2 + a_4 + a_6 = -2 + 6 + 1 = 5$

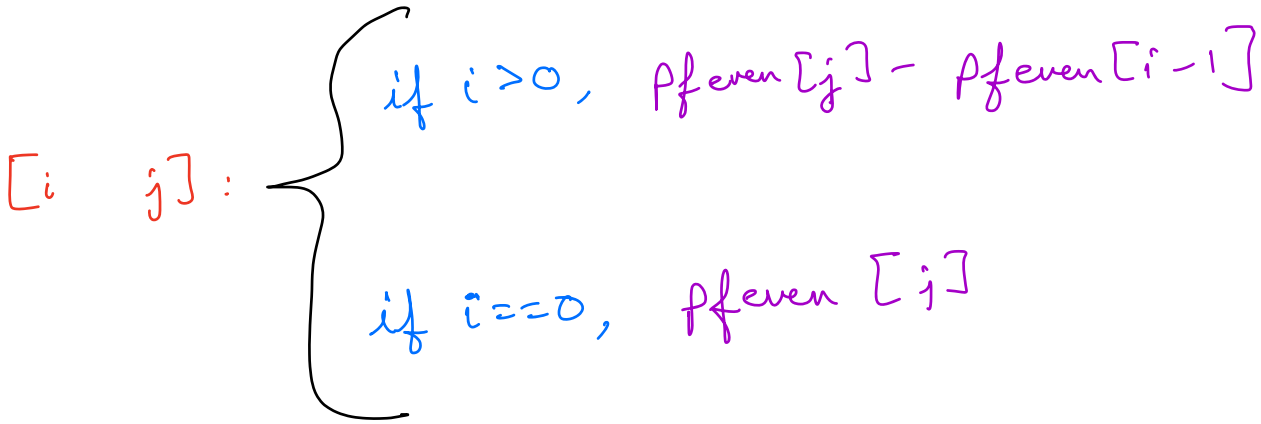
pfeven[6] - pfeven[1]

= 8 − 3 = 5

[3 7]: $a_4 + a_6 = 6 + 1 = 7$

pfeven[7] - pfeven[3-1]

= 8 − 1 = 7



length = N → $\frac{N}{2}$

O(N) → O(N)

Time

Space

Simplicity

Say we delete 4th index. How will the array look like ?

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

Q5 Special Index

HARD

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

Sum of all even index = Sum of all odd index

Count how many special index are there ?

N=6

	0	1	2	3	4	5
A =	4	3	2	7	6	-2

Delete Index 0

	0	1	2	3	4
A =	3	2	7	6	-2

$S_{even} = 8$ $S_{odd} = 8$

Special Index

Delete Index 1

	0	1	2	3	4
A =	4	2	7	6	-2

$S_{even} = 9$ $S_{odd} = 8$

Delete Index 3

	0	1	2	3	4
A =	4	3	2	6	-2

$S_{even} = 4$ $S_{odd} = 9$

Not a special index

Delete Index 4

	0	1	2	3	4
A =	4	3	2	7	-2

$S_{even} = 4$ $S_{odd} = 10$

Not a special index

Delete index 2

	0	1	2	3	4
A =	4	3	7	6	-2

$S_{even} = 9$ $S_{odd} = 9$

Special Index

Not a special index

Delete index 5

	0	1	2	3	4
A =	4	3	2	7	6

$S_{even} = 12$ $S_{odd} = 10$

Not a special index

Count = 2

Brute Force

It will create & return a new array with i^{th} index deleted.

Implement on your own.

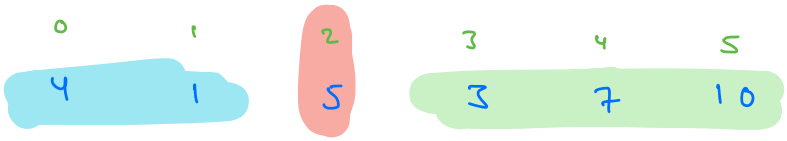
```
int count = 0

for (i=0; i < N; i++) {
    // Need to check if i is
       a special index

    int [] copy = deleteIndex (arr, i)

    int se = 0 , so = 0

    for ( j=0; j <(N-1) ; j++) {

        if ( j is even)
            se += copy[j]

        else
            so += copy[j]

    }

    if ( se == so )
        count ++

}

return count;
```

Time - $O(N^2)$

Space - $O(N)$

Quiz 1

Delete index 2. Sum of all odd indices



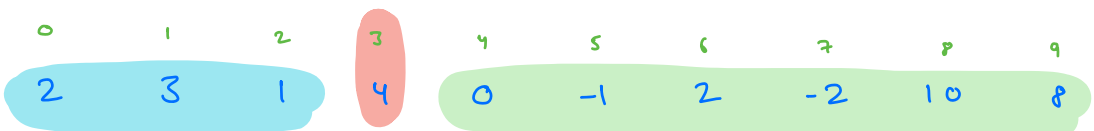
S_{odd} = 1 + 7 = 8

$a_1 + a_4$

sumodd[0 1] + sumeven[3 5]

Quiz 2

Delete index = 3. Sum of all even indices.



S_{even} = 2 + 1 + -1 + -2 + 8 = 8

$a_0 + a_2 + a_5 + a_7 + a_9$

sumeven[0 2] + sumodd[4 9]

Generalisation

S_{even} = sumeven [0 i-1] + sumodd [i+1 n-1]

S_{odd} = sumodd [0 i-1] + sumeven [i+1 n-1]

---

1) Construct pfeven [N] & pfodd [N]

2)

sumeven [0 i-1] = pfeven [i-1]

sumodd [i+1 n-1] = pfodd [N-1] - pfodd [i]

sumodd [0 i-1] = pfodd [i-1]

sumeven [i+1 n-1] = pfeven [N-1] - pfeven [i]

Generic Formula

$$[l \quad r] = \begin{cases} \text{if } l > 0, & pf[r] - pf[l-1] \\ \text{if } l == 0, & pf[r] \end{cases}$$

Pseudocode

```
int specialIndex(int arr[]) {
        n = arr.length

        int pfodd[N], pfeven[N]
        // Fill these prefix sum arrays. - O(N) time

        Count = 0
        for(i=0; i<N; i++) {  -> O(N) time
            // check if i is a special index

            s_even = sumeven[0 i-1] + sumodd[i+1 n-1]
                   = pfeven[i-1] + pfodd[N-1] - pfodd[i]
                     if i==0 : 0

            s_odd  = sumodd[0 i-1] + sumeven[i+1 n-1]
                   = pfodd[i-1] + pfeven[n-1] - pfeven[i]
                     if i==0 : 0

            if ( s_even == s_odd )
                count ++

        }
        return count
}
```

Time = O(N)

Space = O(N)

Doubts

Thank You

Revise Prefix Sum notes before next Session.

If you have solved a problem

→ Check out the editorial.

List

→ Mutable

```
a = [1, 5, 9]
a[2] = 10
```

✓

Tuple

→ Immutable

```
a = (1, 5, 9)
a[2] = 10
```

← Error

→ Slower in performance | → faster to process

Typecasting

```
a = [1, 10, 2]
b = tuple(a)      # (1, 10, 2)

c = (2, 9, 6)
d = list(c)       # [2, 9, 6]
```

Dynamic Arrays

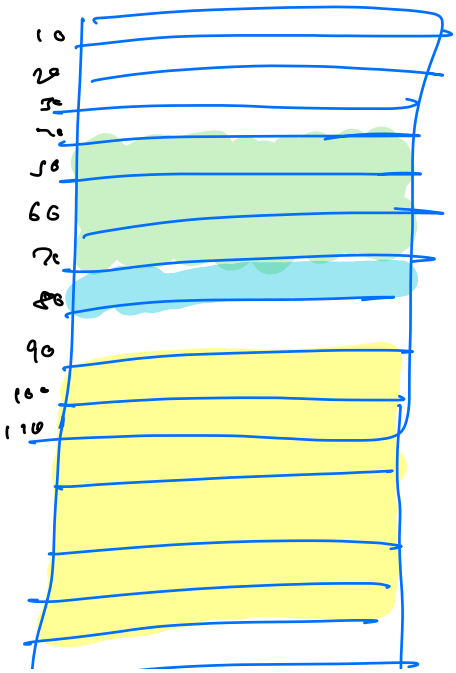
List in Python

```
b = 100
a = [1, 2, 3]
a.append(4)
```

Avg Case Time - O(1)

Worst - O(N)

Contiguous Memory.



Internally,

1) Python will create a new array with size atleast 4.

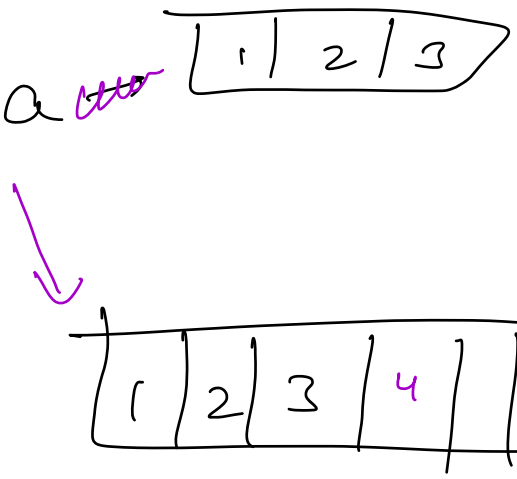
New Size = 6

2) Copy data from old array to new array - O(N)

RAM

3) Then append new data

4) Change the references



5) Garbage collector will free memory of old array.

C++ - vector < > ← Scaler Topics

Java - ArrayList

Good Night

Thank You

Wednesday