Dynamic Arrays

Issue in static arrays? Fixed size

Dynamic Arrays: We need not worry about size

T.C of accessing i^{th} element : O(1)

Java:

```
ArrayList<Integer> arr = new ArrayList<>();
arr.add(50);
arr.add(100);
arr.add(1000);

for(i=0; i<arr.size(); i++){
        print( arr.get(i));
}
```

C++

```
vector<int> a; //vector is created
```

```
a.push_back(60);
//a = {10, 20, 30, 40, 50, 60} after insertion at end
```

```
a.clear(); // a={}
```

```
for(int i=0; i<a.size(); i++) {
    cout<<a[i];
}                              //iterating the vector
```

Python

```
thislist = [] //created list
```

```
thislist.append("orange") //added orange at end
```

```
thislist.clear()    //cleared the list
```

```
for i in range(len(thislist)):
  print(thislist[i])  //iterating on the list
```

Stock Portfolio Performance Tracking

Problem Statement :

Given an array representing the daily profit or loss from a particular stock over a period of days, write a function that calculates the total profit or loss over a given range of days. The function should efficiently handle multiple queries for different ranges without recalculating the sum for each query.

Example:

Stock_Prices[ ] = [-5, 10, 20, 40, 50, -10, 80, -90, -20, -10]
0 1 2 3 4 5 6 7 8 9

Start Day	End	Net Profit /loss
1	3	10 + 20 + 40 = 70
5	8	-10 + 80 - 90 - 20 = -40

Question: Range sum queries

Given an array of size N, and Q queries of the format s(start index) and e (end index), return the sum of elements from s to e

$N \leq 10^6$

$Q \leq 10^6$

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

Q - queries

S	e	Sum
1	3	6+2+4 = 12
2	7	2+4+5+2+8-9 = 12
4	8	5+2+8-9+3 = 9
0	2	-3+6+2 = 5
7	7	-9

queries : [[1,3], [2,7] [4,8] [0,2]]

Brute Force :

For each query, iterate from S to E and find the sum

Pseudocode

```
Function querySum(Queries[][], Array[], querySize, size){
    for(i -> 0 to Queries.length-1){
        L  = Queries[i][0]
        R = Queries[i][1]

        sum = 0
        for( j -> L to R){
            sum += Array[j]
        }
        print(sum)
    }
}
```

Q

N

T.C : $O(Q \cdot N)$

S.C : $O(1)$

$N = 10^6$

$Q = 10^6$

$10^6 \times 10^6 = 10^{12}$ => TLE

Efficient Approach

250-3 / 45 overs

Ex.

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

runs in 7^{th} over = scores[7] - scores[6]

= 65 - 49 = 16

Quiz : Runs from 6^{th} to 10^{th} over

runs : scores[10] - scores[5]

= 97 - 31 = 66

Quiz : 10^{th} over

runs = scores[10] - scores[9] : 97 - 88 = 9

S.....e = scores[e] - scores[s-1]

Quiz: 3rd to 6th

scores[6] − scores[2] = 49 − 8 = 41

Given any over range, we are able to find the runs slored just by subtracting 2 values)

We are able to do this because, after every over we know the cumulative sum from 1st over until current over

Prefix Sum Array

PS[i] : sum of all elys from index 0 i

= a[0] + a[1] + ... a[i-1] + a[i]

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

PS =	2	7	6	13	
	0	1	2	3	4

PS[0] = A[0]

PS[1] = A[0] + A[1]

PS[2] = PS[1] + A[2] = 7 + (-1) = 6

PS[3] : PS[2] + A[3] = 6 + 7 = 13

PS[4] : PS[3] + A[4] = 13 + 1 = 14

PS[i] = PS[i-1] + A[i]

[0.....i) [0.....i-1)

// Construct

```
int PS[N];                    //Declared an array of size N
PS[0] = A[0];
for(i = 1; i < N; i++){
        PS[i] = PS[i-1] + A[i];
}
```

T.C : O(N)

Ex :

	0	1	2	3	4	5
A =	10	32	6	12	20	1
PS :	10	42	48	60	80	81

A =

A:	-3	6	2	4	5	2	8	-9	3	1
	0	1	2	3	4	5	6	7	8	9
PS:	-3	3	5	9	14	16	24	15	18	19
	0	1	2	3	4	5	6	7	8	9

Q queries

S	e	Sum
1	3	PS[3] - PS[0] = 9-(-3) = 12
2	7	PS[7] - PS[1] = 15 - 3 = 12
4	8	PS[8] - PS[3] = 18 - 9 = 9
0	2	PS[2] = 5
7	7	PS[7] - PS[7-1] = 15-24 = -9

Sum(S.....e) = PS[e] - PS[S-1] ✓ (if S!=0)

```
Function querySum(Queries[][], Array[], querySize, size){
    //calculate pf array
    pf[N]
    pf[0] = A[0];
    for(i -> 1 to N-1){
        pf[i] = pf[i-1] + A[i];
    }

    //answer queries
    for( i -> 0 to Queries.length-1){
        L  = Queries[i][0];
        R = Queries[i][1];

        if(L == 0) {
            sum = pf[R]
        }
        else {
            sum = pf[R] - pf[L - 1];
        }

        print(sum);
    }
}
```

O(N)

Q

T.C: (N + Q)

S.C: O(N)

↓

P.S array

Sum of a Range : Prefix Sum Technique

(Product)

(XOR)

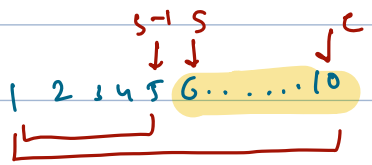
8:32

Question :

Given an array of size N and Q queries with start (s) and end (e) index. For every query, return the sum of all **even indexed elements** from **s to e**.

Example

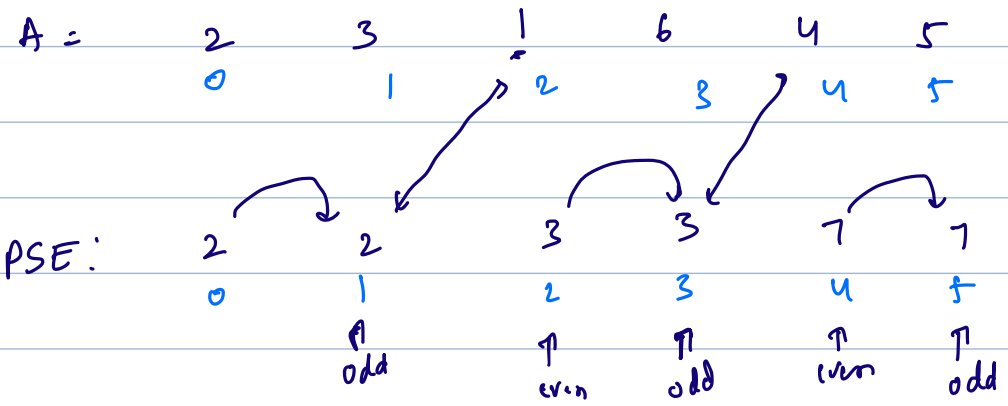
```
          0  1  2  3  4  5
A[ ] = { 2, 3, 1, 6, 4, 5 }
Query :        Ans
 1 3           A[2] = 1
 2 5           A[2]+A[4] = 1+4 = 5
 0 4           A[0]+A[2]+A[4] = 2+1+4 = 7
 3 3           0
```



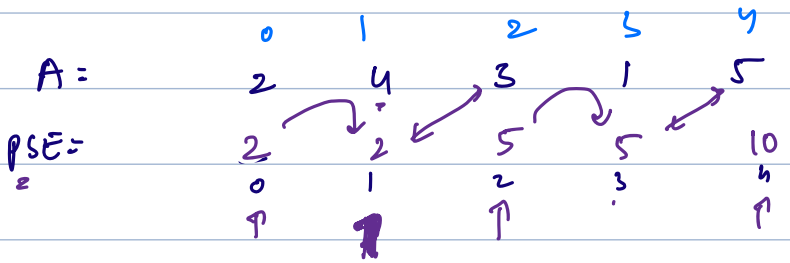
Efficient Approach

Construct PS array for even indexed elements

PSE[i] = Sum of even indexed elements from index 0 to index i




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



S	E	Sum
1	3	PSE[3] − PSE[1−1] = 3
0	4	PSE[4] = 10
3	3	PSE[3] − PSE[2] = 5 − 5 = 0

```
Function sumOfEvenIndexed(Array[], Queries[][], N){
    // prefix sum for even indexed elements
    PSe[N];

    PSe[0] = Array[0]

    for(i -> 1 to N-1){
        if(i % 2 == 0){
            PSe[i] = PSe[i-1] + Array[i];
        }
        else {
            PSe[i] = PSe[i-1];
        }
    }
    for(i -> 0 to Queries.length-1) {
        s = Queries[i][0]
        e = Queries[i][1]
        if(s == 0){
            print(PSe[e])
        }
        else {
            print(PSe[e]-PSe[s-1])
        }

    }

}
```

Construct PSE array
O(N)

O(Q)

T.C: O(N+Q)

S.C: O(N)
↳ PSE array

Question: Sum of odd indexed elements

Steps

1) Construct PSO array

PSO[i] = sum of odd indexed eles from 0 to i

```
for (                    ) {
    if (i%2 == 1) {
        PSO[i] = PSO[i-1] + A[i];
    }
    else {
        PSO[i] = PSO[i-1];
    }
}
```

Question:

Given an array of size N, count the number of special index in the array.
Note: Special Indices are those after removing which, sum of all **EVEN** indexed elements is equal to sum of all **ODD** indexed elements.

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

Remove

i = 0 → 3 2 7 6 -2 → $S_o = 8$ $S_e = 8$ ✓
0 1 2 3 4

i = 1 → 4 2 7 6 -2 → $S_o = 9$ $S_e = 9$ ✗
0 1 2 3 4

i = 2 → 4 3 7 6 -2 → $S_o = 9$ $S_e = 9$ ✓
0 1 2 3 4

Quiz: Sum of odd indexed

⇒ A = 4 1 3 7 10
0 1 2 3 4

Sum of even index

A : 4 1 7 10 ✓
0 1 ~~2~~ 3 sum of odd index

Ans : $S_{odd}[0, 1] + S_{even}[3, 4]$

Quiz: Sum of odd indexed elements

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

Ans: $S_{odd}[0, 2] + S_{even}[4, 9]$

3 + (0 + 2 + 10) = 15

Quiz: Sum of even indexed

A = | 2 | 3 | 1 | 4 | 0 | -1 | 2 | -2 | 10 | 8 | ← odd
0 1 2 3 4 5 6 7 8 9

A' = | 2 | 3 | 1 | 0 | -1 | 2 | -2 | 10 | 8 | → even
0 1 2 3 4 5 6 7 8

$S_{even} = S_{even}[0, 2] + S_{odd}[4, 9]$

A = A_0 A_1 A_2 . . . . . A_{i-1} A_i A_{i+1} . . . . . A_{N-1}

A' =

After removing index i,

S_{odd} = $S_{odd}[0, i-1]$ + $S_{even}[i+1, N-1]$

$PSO[i-1]$ + $PSE[N-1] - PSE[i]$

S_{even} = $S_{even}[0, i-1]$ + $S_{odd}[i+1, N-1]$

$PSE[i-1]$ + $PSO[N-1] - PSO[i]$

$S_{odd}[0, i-1] = PSO[i-1]$; ✓ => i != 0

$S_{even}[0, i-1] = PSE[i-1]$; => i != 0

$S_{even}[i+1, N-1] = PSE[N-1] - PSE[i]$;

$S_{odd}[i+1, N-1] = PSO[N-1] - PSO[i]$;

Pseudocode

```
Function count_special_index(arr[], n)
{
    // prefix sum for even indexed elements
    PSe[n];                                          O(N)
    // prefix sum for odd indexed elements
    PSo[n];                                          O(N)

    //Say we have already calculated PSe and PSo

    //Code to find Special Indices
    count = 0;
    for (i -> 0 to n-1) {

        if(i==0) {
            Se = PSo[n-1] - PSo[i]; //sum from [i+1 n-1]
            So = PSe[n-1] - PSe[i]; //sum from [i+1 n-1]
        }
        else {
            Se = PSe[i-1] + PSo[n-1] - PSo[i]; //sum even from [0 to i-1] and odd from [i+1 n-1]
            So = PSo[i-1] + PSe[n-1] - PSe[i]; //sum odd from [0 to i] and even from [i+1 n-1]
        }

        if(Se == So) {
            count++;
        }

    }

    return count;
}
```

O(N)

T.C.: O(N) + O(N) + O(N) = O(N)

S.C.: O(N) + O(N) = O(N)
PSE PSO

Sold :