Q1 (Easy) Find an equilibrium index of an array.

An index i is said to be equilibrium index, if

Sum of all array elements before i^{th} index = Sum of all array elements after i^{th} index

$$\sum_{j=0}^{i-1} arr[j] = \sum_{j=i+1}^{N-1} arr[j]$$

Note: $[0, i-1]$ & $[i+1, N-1]$

Convert the equilibrium index

Ex: $arr[] = \{-7, 1, 5, 2, -7, 3, 0\}$

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

Equi index

$i=3$: $[0, 2] = -1 = [4, 6] = -1$ ✓

$i=4$: $[0, 3] = -4 = [5, 6] = 1$ ✗

Equi index

$i=2$: $[0, 1] = 4$, $[3, 6] = 4$ ✓

$i=4$: $[0, 3] = 3$, $[5, 6] = 3$ ✓

Ex: $arr[] = \{3, -1, 2, -1, 1, 2, 1\}$

Output: 2

Solution:

Brute Force approach

For every index check equilibrium or not.

```
cnt = 0
i = 1, i < N-2, i++{
    check 1st index cnt:
        cnt = cnt + arr[0]
        sum [i+1, N-1]
        cnt++
}
```

TC: $O(N^2)$ SC: $O(1)$

Note: Sum $[0, i-1] + arr[i] + Sum[i+1, N-1] = Sum[0, N-1]$

Sum $[i+1, N-1] = PF[N-1] - PF[i]$

Sum $[0, i-1] = PF[i-1]$

Step 1: Construct PF[]

```
cnt = 0
i = 1, i < N-2, i++{
    check 1st index cnt:
        leftSum = PF[i-1]
        RightSum = PF[N-1] - PF[i-1] - arr[i]
        if (leftSum == RightSum){
            cnt++
        }
}
```

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

Sol2: // LS = Left Sum, // RS = Right Sum

LS = arr[0]

Initialize on array calculate Total_Sum = N

$i=1$: $arr[1]$

$i=2$: $arr[0] + arr[1]$

$i=3$: $arr[0] + arr[1] + arr[2]$

```
i = 1, i < N-2, i++{
    RS = Total_Sum - arr[i] - LS
    if ( LS == RS ) { cnt++ }
    LS += arr[i] // update LS
}
```

TC: $N + N-2 \approx O(N)$

SC: $O(1)$

2) Count Special Elements

Given N array elements, count number of special elements.

An element is special if after removing the element from the array Sum of all Elements at odd index positions = Sum of all Elements at Even index positions

Todo:

Without extra space

Step 1: Calculate EB & OB in $O(N)$

Step 2: Calculate Sum of all Even & Sum of Odd

Ex: 2 4 2 3 3

Deleting 0 (Special Index)

Sum Even : 5

Sum Odd : 5

Deleting 2:

Sum E = 9

Sum O = 9

Delete 0

Sum E = 8

Sum O = 8

Delete 3

Sum E = 4

Sum O = 7

Sol: Check if every index is special or not

```
c = 0
i = 0, i < N, i++{
    check if i is special:
        Sum_Even_before = PFE[i-1]
        Sum_Odd_before = PFO[i-1]
        Sum_Even_after = EA
        Sum_Odd_after = OA
        if ( EB+OA == OB+EA ){
            c(++)
        }
}
return c
```

Sol

PFE[i] = Sum of all Even index from [0, i]

PFO[i] = Sum of all Odd index from [0, i]

PFE[N], PFO[N]

	0	1	2	3	4	5	6	7	8
arr[] :	3	4	2	7	1	5	6	9	10
PFE[] :	3	4	5	5	14	14	20	20	30
PFO[] :	0	4	4	11	11	10	10	19	19

PFE[0] = arr[0], PFO[0] = 0

```
i = 1, i < N, i++{
    if ( i % 2 == 0 ){
        PFE[i] = PFE[i-1] + arr[i]
        PFO[i] = PFO[i-1]
    }
    else {
        PFO[i] = PFO[i-1] + arr[i]
        PFE[i] = PFE[i-1]
    }
}
```

```
i = 0, i < N, i++{
    Even_before = EB, OB, EA, OA
    if i == 0 change
    EB = 0, OB = 0
    add_Bef:
        EB += PFE[i-1]
        OB += PFO[i-1]
    Find_After:
        EA = Sum of all Even index [i+1, N-1]
        = PFE[N-1] - PFE[i]
        OA = Sum of all Odd index [i+1, N-1]
        = PFO[N-1] - PFO[i]
    if ( EB + OA == OB + EA ){
        c(++)
    }
}
```

TC: $O(N+N) = O(2N) = O(N)$ SC: $O(N)$

Even_Before [i-1] → Odd index value

Odd_Before [i-1] → Odd_After [i]

Find_After [i] → Odd_After [i]

Sum [i+1, N-1] = PFE[N-1] - PFE[i] → Sum of Even Index [0, N-1] [0, i]

→ 100 % 2 == 0 ??

Q3 Given an array of N Times, find min subarray sum with size k

Ex: 7 5 3 6 2 -1 10 4 7

N = 9

Contain zero + one array

→ $O(N)$

Sol: For all subarrays of size k, get min sum

k = 4

	Original		
[0, 3]	[0, k-1] = 2	s = 0	k+0
[1, 4]	[1, k] = s = s - arr[0] + arr[k]	s = 2	k+1
[2, 5]	[2, k+1] = s = s - arr[1] + arr[k+1]	s = 3	k+2
[3, 6]	[3, k+2] = s = s - arr[2] + arr[k+2]	s = 4	k+3
[-e]	[i, k+i] = s = s - arr[i-1] + arr[k+i]		

i = 1

Sub	
0	N-k
1	N-1
2	N-2
3	N-3
k	N-k+1
N	1

→ $O(N-k+1)$

Step 1: s = 0

```
i = 0, i < k, i++{
    s = s + arr[i]
}
ans = s
i = 0, i < N-k, i++{
    s = s + arr[i] - arr[i-1]
    ans = min(ans, s)
}
```

TC: $O(N)$ SC: $O(1)$

```
i = k : (i, k) . s = s + arr[i] - arr[i-k]
```

[N-k, N-2] → s2

(N-k, N-1) → s = s - arr[N-k] + arr[N-1]

```
i = 0, i < N-k, i++{
    s = s - arr[i] + arr[i+k]
    ans = min(ans, s)
}
```

Q4 Given an array of size N, find the number of distinct elements in every subarray of size k.

k = 4

Ex: 4 1 3 1 5 2 5 6 7 8

Step 1: Brute force approach

for every subarray of size k, insert in hashset & get size.

$\binom{N-k+1}{k}$ = pick one k element in set

Become TC: $(N-k+1) * (k)$

SC: $O(N)$

→ $(N-k+1)(k)$, $k = N/2$

$(N-N/2+1)(N/2)$

→ $(N/2+1)(N/2)$

TC: $O(N^2)$

Output: 3 3 4

Using deleting an element

→ will take all operations

Use Hashmap

<arr[i], freq>

k = 4 0 1 2 3 4 5 6 7 8

Ex: 4 1 3 1 5 2 5 6 7

Step 1: <key, freq>

<3, 1>

<1, 2>

<5, 1>

<2, 1>

Output: 3 3 4

```
i = 0, i < k, i++{
    if ( arr[i] ∈ hm ){
        hm[arr[i]]++
    }
    else {
        hm[arr[i]] = 1
    }
}
print(hm.size())
i = k, i < N, i++{
    // Add arr[i], Delete arr[i-k]
    if ( arr[i] ∈ hm ){
        hm[arr[i]]++
    }
    else { hm[arr[i]] = 1 }
    hm[arr[i-k]]--
    if ( hm[arr[i-k]] == 0 ){
        hm.remove(arr[i-k])
    }
    print(hm.size())
}
```

[i-k] : Delete arr[i], add arr[i]

if freq == 0, Remove entire key

[i-k] : Delete arr[i], add arr[i]

Bookkeeping:

hashmap<int, int> hm;

TC: $O(k)$

Index arr[i]

Remove arr[i-k]

TC: $O(N/k)$

Overall

TC: $O(N)$ SC: $O(k)$

Josephus Generalized Case

k = 4

$(N, k) = \left(\binom{N-1}{N}k - 1\right) \% N + 1$

[0, N-1]

5 ✗

[0, k-1]