Important Problems

!! Announcement !!

Contest-2 this friday on regular time

Contest reattempt till Sunday 11:59 PM

Topics : Bit Manipulation , Maths and Sorting

Question 1

Given an array of size N & Q queries of type [s,e]. for every query return the sum of all even indexed elements in the range from s to e. ⇒ $0 <= s <= e < n$

eg

	2	3	1	6	4	5
	0	1	2	3	4	5

Q=4

s	e	sum
1	3	1
2	5	1+4 = 5
0	4	2+1+4 = 7
3	3	0 [no even index]

Idea 1 : for each query, [TODO coding part]
iterate from s to e and sum even-indexed elements.

TC : $O(N*Q)$

SC : $O(1)$

Idea 2 : Prefix Sum

$$pf[i] = \underbrace{a[0] + a[1] + \cdots + a[i-1]}_{\downarrow} + a[i]$$

$$pf[i] = pf[i-1] + a[i]$$

pfe → prefix sum of even-indexed elements

$pfe[i] = pfe[i-1] + a[i]$ (if $i\%2 == 0$)

$pfe[i] = pfe[i-1]$ (if $i\%2 == 1$)

	0	1	2	3	4	5
A :	2	3	1	6	4	5
pf :	2	5	6	12	16	21
pfe :	2	2	3	3	7	7

Code

```
pfe[n]

pfe[0] = a[0]

for(i=1; i<n; ++i) {     → N iterations
    if (i%2 == 0)
        pfe[i] = pfe[i-1] + a[i]
    else
        pfe[i] = pfe[i-1]
}

for(i=0; i<Q; ++i) {     → Q iterations
    // s,e index
    if(s==0)
        print( pfe[e] )
    else
        print( pfe[e] - pfe[s-1] )
}
```

TC : O(N+Q)

SC : O(N)

A:	2	4	3	1	5
	0	1	2	3	4
pfo :	0	4	4	5	5

↳ prefix sum of odd-indexed elements

```
pfo[0] = 0

for (i=1; i<n; ++i) {
    if (i%2 == 1)
        pfo[i] = pfo[i-1] + a[i]
    else
        pfo[i] = pfo[i-1]

}
```

Question 2 — Special Index

Given an array, count no. of special index in array.

→ An index is special, if after removing the index :

Sum of all even-indexed elements = Sum of all odd-indexed elements

eg A:	4	3	2	7	6	-2
	0	1	2	3	4	5

i						Se	So	
0	3	2	7	6	-2	8	8	✓
	0	1	2	3	4			
1	4	2	7	6	-2	9	8	✗
	0	1	2	3	4			

2 4 3 7 6 -2 9 9 ✓

4	3	7	6	-2
0	1	2	3	4

Idea 1 : for each index,
create a new array after removing the index, calculate S_e & S_o and compare.

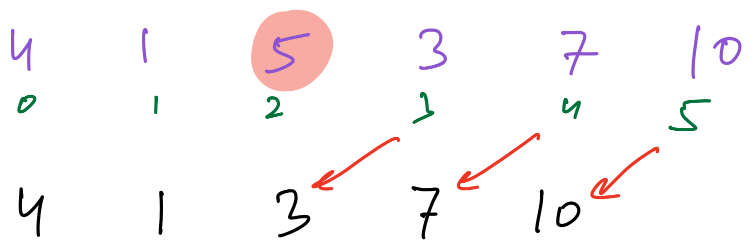
Code

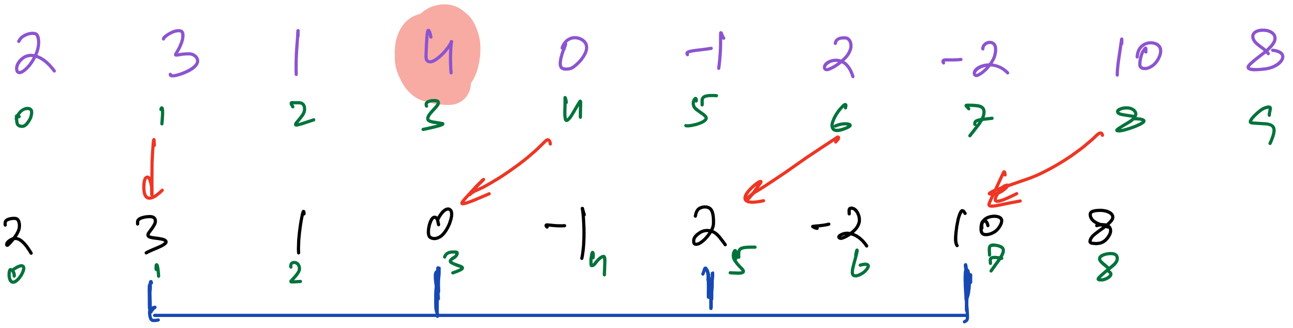
```
int specialCount(a[]) {
    ans=0
    for (i=0; i<n; ++i) {   → N iterations
        // remove i index
        // a[n] = a0, a1, . . . . . ., ai-1, ai, ai+1, . . . , an-1
        // create new array w/o a[i]
        temp[n-1] =

        // iterate on temp and calculate Se & So - TODO
                                                     TC: O(N)
        if (Se == So)
            ++ans
    }
    return ans.
}
```

TODO
TC: O(N)
SC: O(N)

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

Observation Time





↳ odd index values after delete

Sum of odd-indexed elements after removing index 3 =

Sum of odd-indexed elements from [0-2]
3

+

sum of even-indexed elements from [4-9]
0 + 2 + 10 = 12

in the original array

Sum of even-indexed elements after removing index 3 =

Sum of even-indexed elements from [0-2]

+

Sum of odd-indexed elements from [4-9]

in the original array

After removal of index i

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

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

We have already solved prefix sum for even and odd indexed elements.

pfe → prefix sum of even-indexed elements

pfo → prefix sum of odd-indexed

$Se[0, i-1] = pfe[i-1]$

$So[0, i-1] = pfo[i-1]$

$Se[i+1, n-1] = pfe[n-1] - pfe[i]$

$So[i+1, n-1] = pfo[n-1] - pfo[i]$

Code

```
Create pfe[n] & pfo[n]  → TODO → O(N)
ans = 0
for (i=0; i<n; ++i) {  → N iterations
   // Se = pfe[i-1] + (pfo[n-1] - pfo[i])
   // So = pfo[i-1] + (pfe[n-1] - pfe[i])

   if (i==0) {
      Se = pfo[n-1] - pfo[i]
      So = pfe[n-1] - pfe[i]
   }
   else {
      Se = pfe[i-1] + pfo[n-1] - pfo[i]
      So = pfo[i-1] + pfe[n-1] - pfe[i]
   }
}
```

```
    if (Se == So)
        ++ans
  }
  return ans
}
```

TC : O(N)

SC : O(N)

Question 3 - Majority Element

Given an array, return if there exists an element with frequency > N/2. (N = array length)

eg $a[6]$ = 1 2 1 6 1 1

freq(1) = 4 $N/2 = 6/2 = 3$ ⇒ $4 > 3$ **YES**

1 is majority elent

eg $a[9]$ = 3 4 4 8 4 9 4 3 4

freq(4) = 5 $N/2 = 9/2 = 4$ ⇒ $5 > 4$ YES

eg $a[10]$ = 4 6 5 3 4 5 4 4 4 8

freq(4) = 5 $N/2 = 10/2 = 5$ ⇒ $5 \not> 5$ NO

Ideas: Count freq. of each element & compare with N/2.

1. Using 2 nested loops → $TC: O(N^2)$ $SC: O(1)$

2. Hashmap/Dictionary → $TC: O(N)$ $SC: O(N)$

3. Sort the array

   $a(7) = \{1\ 3\ 1\ 1\ 3\ 2\ 1\ 1\}$ $TC: O(N\log N)$

   $sort(a) = \{1\ 1\ 1\ 1\ 1\ 2\ 3\ 3\}$ $SC: O(1)$

Ignore if not understood

At max how many majority elements can be there in array? ⇒ $\boxed{Ans = 1}$

assume 2 majority elements x & y

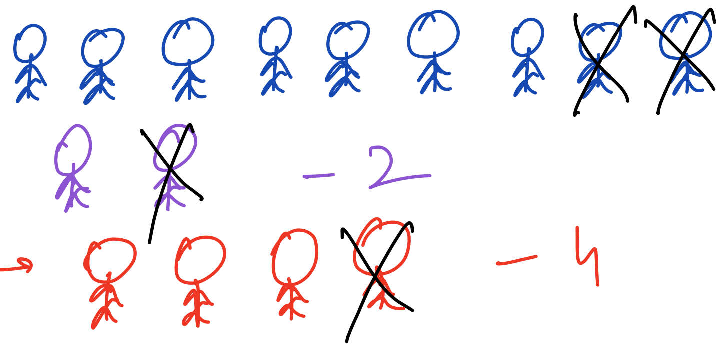
$$freq(x) > N/2$$

$$freq(y) > N/2$$

$$\boxed{freq(x) + freq(y) > N}$$

→ invalid because we only have N elements.

Election → 15 MLAs $15/2 = 7$



Rohit → – 9

Dileep → – 2

Akansha → – 4

Rohit → 9 $15/2 = 7$ $9 > 7$

↓ 2 disquality

Rohit → 8 $13/2 = 6$ $8 > 6$

↓ 2 disquality

Rohit → 7 $11/2 = 5$ $7 > 5$

Observation :

If you delete two distinct elements , majority won't change.

$a[9] = \cancel{5}\ \cancel{4}\ \cancel{4}\ \cancel{3}\ \boxed{4}\ \cancel{2}\ \cancel{4}\ \cancel{3}\ \cancel{4}$

$freq(4) = 5$ $9/2 = 4$ $5 > 4$

a[11] = 3 ~~3~~ ~~4~~ ~~6~~ ~~1~~ ~~3~~ ~~2~~ ~~5~~ ~~3~~ 3 ~~3~~ ~~1~~

freq(3) = 6 $11/2 = 5$ $6 > 5$

Implementation

1. Assume first element as majority element
2. If you get same element, increase freq.
3. If you get different element, decrease freq.
4. If freq = 0, change majority element.

a[11] = 3 3 4 6 1 3 2 5 3 3 3

ele = 3 | ele = 1 | ele = 2 | ele = 3
freq = ~~1~~ ~~2~~ ~~1~~ 0 | freq = ~~1~~ 0 | freq = ~~1~~ 0 | freq = ~~1~~ ~~2~~ 3

freq(3) = 6
$11/2 = 5$

a[10] = 3 3 4 6 1 3 2 5 8 8

ele = 3 | ele = 1 | ele = 2 | ele = 8
freq = ~~1~~ ~~2~~ ~~1~~ 0 | freq = ~~1~~ 0 | freq = ~~1~~ 0 | freq = ~~1~~ 2

freq(8) = 2
$10/2 = 5$

If final element freq $\not> N/2$ then

NO MAJORITY

Code

```
int majority ( a[] ) {
    n = a.length
    ele = a[0] , freq = 1
    for ( i = 1; i < n; ++i ) {
        if ( freq == 0 ) {
            ele = a[i] , freq = 1
        }
        else if ( ele == a[i] )
            ++freq
        else
            --freq

    }
    c = 0
    for ( i = 0; i < n; ++i ) {
        if ( a[i] == ele )
            ++c
    }
    if ( c > n/2 )   return ele
    else    return   NO_MAJORITY
```

TC : O(N)

SC : O(1)

}

Moore's Voting Algo

8 8 1 1 1 1 1 1 5

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

1 1 1 3 1 1 3 5 4

ele = 1

freq = ~~3~~ ~~2~~ ~~4~~ 1

Homework

Given array return if there exist an element with freq > N/3   SC : O(1)

Hint: Previously we deleted 2 different elements.

Here, you have to delete 3 different elements.

So, you have to keep 2 variables : ele_1 & ele_2 .