Today's Quote →

IF YOU GET TIRED
LEARN TO REST,
NOT TO QUIT.

Content:
↳ 2 Google Problems

arr →

5	7	4	-3	2	8
0	1	2	3	4	5

pEven →

5	5	9	9	11	11
0	1	2	3	4	5

podd →

0	7	7	4	4	12

pEven[i] → sum of even indexed elements upto i^{th} index

podd[i] → " " odd " " " " " .

Q1:) Special Index → {Hard}

An index is said to be special index, if after deleting index, $\boxed{\text{Sum of all even index elements} = \text{Sum of all odd index elements.}}$

Count total no. of special indices.

Eg: arr[6] : [4 3 2 7 6 -2]
$\quad$ 0 1 2 3 4 5

delete 0th. index

cp → [3 2 7 6 -2]
$\quad$ 0 1 2 3 4

$S_e = 3+7+(-2) = 8$
$S_o = 2+6 = 8$

delete 1st index

cp → [4 2 7 6 -2]
$\quad$ 0 1 2 3 4

$S_e = 4+7+(-2) = 9$
$S_o = 2+6 = 8$

delete 2nd index

C → [4 3 7 6 -2]
$\quad$ 0 1 2 3 4

$S_e = 4+7+(-2) = 9$
$S_o = 3+6 = 9$

delete 3rd index

Cp → [4 3 2 6 -2]
$\quad$ 0 1 2 3 4

$S_e = 4+2+(-2) = 4$
$S_o = 3+6 = 9$

delete 4th index

Cp = [4 3 2 7 -2]
$\quad$ 0 1 2 3 4

$S_e = 4+2+(-2) = 2$
$S_o = 3+7 = 10$

delete 5th index

Cp = [4 3 2 7 6]
$\quad$ 0 1 2 3 4

$S_e = 12$, $S_o = 10$

idea: for every index i, create cp[N-1] where arr[i] is removed. In this cp[], calculate S_e and S_o
if (S_e == S_o) increment count.

pseudo-code

```
void specialIndices ( arr, N) {
    count = 0
    for( i = 0 ; i < N ; i++) {
        //create cp[N-1] where arr[i] is removed
        //arr[] → [ 0  1  2 -- i-1  i  i+1 -  N-2  N-1 ]   → N
        // cp[N-1] → [ copy these        copy these ]
        iterate and calculate Se & So in cp[]   → N
        if( Se == So) { count ++ }
    }
    return count ;
}
```

{Todo}

$T.C \rightarrow O(N^2)$
$S.C \rightarrow O(N)$

idea 2 :

$N = 10$

arr[10] :

3	2	6	8	2	9	7	6	4	12
0	1	2	3	4	5	6	7	8	9

odd → even
even → odd.

Delete index 4

C_p →

3	2	6	8	9	7	6	4	12
0	1	2	3	4	5	6	7	8

$S_e = C_p S_e[0-8] = C_p S_e[0-3] + C_p S_e[4-8]$

[After deleting 4th index $= arr\ S_e[0-3] + arr\ S_o[5-9]$]

arr[12] :

2	1	3	0	6	7	3	4	5	6	10	2
0	1	2	3	4	5	6	7	8	9	10	11

even → odd
odd → even

Delete 5th index

C_p →

2	1	3	0	6	3	4	5	6	10	2
0	1	2	3	4	5	6	7	8	9	10

$S_e = C_p S_e[0-10] = C_p S_e[0-4] + C_p S_e[5-10]$

After deleting 5th index $= arr\ S_e[0-4] + arr\ S_o[6-11]$

$S_o = C_p S_o[0-10] = C_p S_o[0-4] + C_p S_o[5-10]$

$= arr\ S_o[0-4] + arr\ S_e[6-11]$

// Generalisation

$$arr[N] : \begin{array}{ccccccccccc} [a_0 & a_1 & a_2 & a_3 & ---- & a_{i-1} & a_i & a_{i+1} & --- & a_{N-2} & a_{N-1}] \\ 0 & 1 & 2 & 3 & & i-1 & i & i+1 & & N-2 & N-1 \end{array}$$

Deleting ith index

$$C_p \rightarrow \begin{array}{ccccccccc} [a_0 & a_1 & a_2 & a_3 & --- & a_{i-1} & a_{i+1} & a_{i+2} & --- & a_{N-1}] \\ 0 & 1 & 2 & 3 & & i-1 & i & i+1 & & N-2 \end{array}$$

$$S_e = C_p S_e[0-(N-2)] = C_p S_e[0-(i-1)] + C_p S_e[i-(N-2)]$$

S_e after deleting ith_index element	$= arr\ S_e[0-(i-1)] + arr\ S_o[(i+1)$ to $(N-1)]$
S_o after deleting ith_index element	$= arr\ S_o[0-(i-1)] + arr\ S_e[(i+1)$ to $(N-1)]$

$\{sum(i,j) = pSum[j] - pSum[i-1]\}$

Final observations

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

$$\{ S_e = \underline{pEven[i-1]}^{*} + pOdd[N-1] - pOdd[i] \}$$

$$\{ S_o = \underline{pOdd[i-1]}^{*} + pEven[N-1] - pEven[i] \}$$

$*\{i \neq 0\}$

$$\left[\begin{array}{l} sum(0, i-1) = pSum[i-1] \\ sum(i+1, N-1) = pSum[N-1] - pSum[i]. \end{array} \right]$$

pseudo code →

```
int specialIndices ( arr, N ) {
    // Create pEven[N], podd[N] {Todo}
    count = 0
    for ( i = 0 ; i < N ; i++ ) {
        // we need to delete ith index.
        Se = podd[N-1] - podd[i]
        if ( i != 0 ) { Se += pEven[i-1] }
        So = pEven[N-1] - pEven[i]
        if ( i != 0 ) { So += podd[i-1] }
        if ( Se == So ) { count ++ }
    }
    return count
}
```

2N iterations
2N space

N iterations.

T.C → O(N)
S.C → O(N)

→ [Pick from Both Sides.]

Q2) Given N +ve elements. Find majority element.

$\underbrace{\text{majority element}}$ → An element with freq $> N/2$.

Eg1: arr[6] : { 1 2 1 6 1 1 }

index: 0 1 2 3 4 5

$4 > 6/2$ $\boxed{ans = 1}$

Eg2: arr[9] : { 3 4 4 8 4 9 4 3 4 }

index: 0 1 2 3 4 5 6 7 8

$5 > 9/2$ $\boxed{ans = 4}$

Eg3: arr[12] : { 3 3 4 6 1 3 2 5 3 3 3 3 }

index: 0 1 2 3 4 5 6 7 8 9 10 11

$f(3) = 7 > \frac{12}{2}$ $\boxed{ans \to 3}$

Q.) arr[10] : { 4 3 3 3 4 3 4 4 4 3 }

index: 0 1 2 3 4 5 6 7 8 9

$f(4) = 5 > 10/2$

$f(3) = 5 > 10/2$

Ideas :

① For every element, find how many times it is present in the array.

Nested loops → $T.C = O(N^2)$, $S.C \to O(1)$

② Sort the array. $[T.C \to O(N \log N)]$

arr → { 3 5 1 3 7 3 3 3 5 }

arr → { 1 3 3 3 3 3 5 5 7 }

Observations : arr[N].

At max how many majority elements can we have?

// Let's say we have 2 majority elements $\to m1, m2$

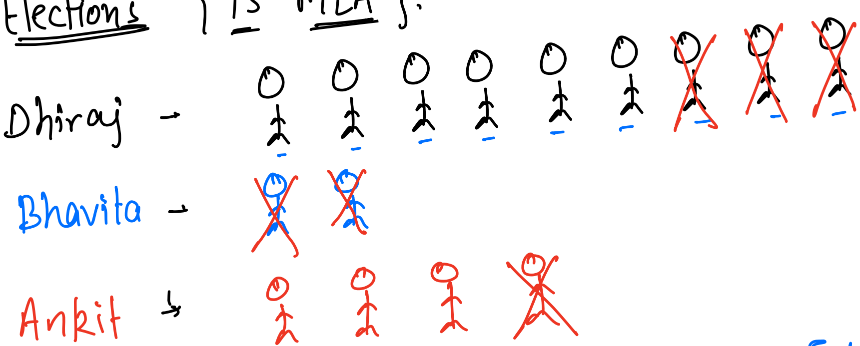
$$freq(m_1) > N/2$$

$$freq(m_2) > N/2$$

$$\boxed{freq(m_1) + freq(m_2) > N}$$, But size of arr is N.

$\therefore$ By contradiction, at max only 1 majority element is possible.

Elections {15 MLA}.



Dhiraj : $9 > 15/2$ [Yes].

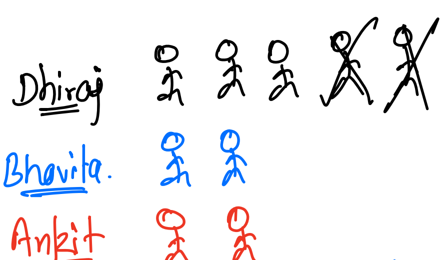
Dhiraj : $8 > 13/2$ [Yes].

Dhiraj : $7 > 11/2$ [Yes].

Dhiraj : $6 > 9/2$ [Yes].

observation :

[When we delete 2 distinct items, majority doesn't change.]



obs → [when we delete 2 - identical items, majority can change] arr → { $\cancel{4}_0$ $\cancel{4}_1$ $\cancel{3}_2$ $\cancel{8}_3$ $\cancel{8}_4$ $\cancel{4}_5$ $\cancel{9}_6$ $\cancel{4}_7$ 4_8 }

freq(4) = 5 > 9/2

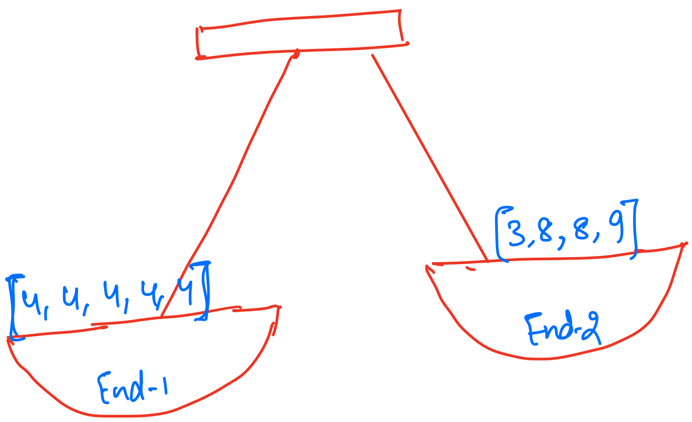
freq(4) = 4 > 7/2 : Yes.

freq(4) = 3 > 5/2 : Yes.

freq(4) = 2 > 3/2 : Yes.

freq(4) = 1 > 1/2 : Yes.

[How can we delete 2 distinct elements always ?]



[Weight balancer]

Assm → weight of every element is 1 Kg.

[4, 4, 3, 8, 8, 4, 9, 4, 4].

arr → { 4 | 4 | 3 | 8 | 8 | 4 | 9 | 4 | 4 }

0 1 2 3 4 5 6 7 8

element = ~~4~~ ~~8~~ ~~9~~ 4

freq = ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

⇓

[no majority element]

arr → { 4 | 2 | 5 | 2 | 7 | 4 | 4 }

0 1 2 3 4 5 6

[ele = ~~4~~ ~~5~~ ~~7~~ 4] ⇒ [potential candidate for my ans.]

freq = ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

⇒ [No majority element]

pseudo code:

```
int majority Element ( arr, N) {
    ele = arr[0] , freq = 1
    for( i = 1 ; i < N ; i++) {
        if ( freq == 0) {
            ele = arr[i] , freq = 1
        }
        else if ( arr[i] == ele ){
            freq++
        }
        else {
            freq--;
        }
    }
    count = 0
    for( i = 0 ; i < N ; i++) {
        if (arr[i] == ele) { count++ }
    }
    if (count > N/2) { return ele }
    else { "No majority element" }
                  [-1]
}
```

Moore's Voting Algorithm

T.C → O(N)

S.C → O(1)

arr → { 4 | 3 | 4 | 4 | 7 | 5 | 4 | 4 | 8 }.

element = 4 → [potential candidate]

freq = ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ 1

freq(4) ⇒ 5 > 9/2 "4 will be the majority element"

arr = { 4 | 4 | 4 | 8 | 8 | 8 | 8 | 8 | 3 }
0 1 2 3 4 5 6 7 8

element = ~~4~~ 8

freq = ~~1~~ ~~2~~ ~~3~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ 1

arr → { 4 2 5 2 7 4 4 6 7 }
0 1 2 3 4 5 6 7 8

{ 1 2 2 2 4 5 6 7 8 }

element =

$> N/2 \Rightarrow$ deleting 2 distinct elemts.

$> N/3 \Rightarrow$ " 3 distinct elements.

arr = { 2 3 4 2 2 5 3 3 5 7 }

ele = ~~2~~ ~~4~~ ~~2~~ 3

freq = ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~1~~ 0

{ a b c d e f g g g g }

element = g

Pick From Both Sides.

B = 4.

arr →

7	3	-1	4	3	9	1
0	1	2	3	4	5	6

left	right		
0	4	→	16
1	3	→	19
2	2	→	20
3	1	→	10
4	0	→	13.

$\boxed{ans = 20}$

S.C → O(1)

lsum = 0 , rsum = 0

// iterate & calculate sum of last B elements.

after this, rsum = 16. , maxSum = rsum.

```
for( i = 0; i < B; i++){
    lsum += arr[i]
    rsum -= arr[N-B+i];

    if(lsum + rsum > maxSum){
        maxSum = lsum + rsum
    }
}
return maxSum;
```

S.C → O(1)

T.C → O(B)