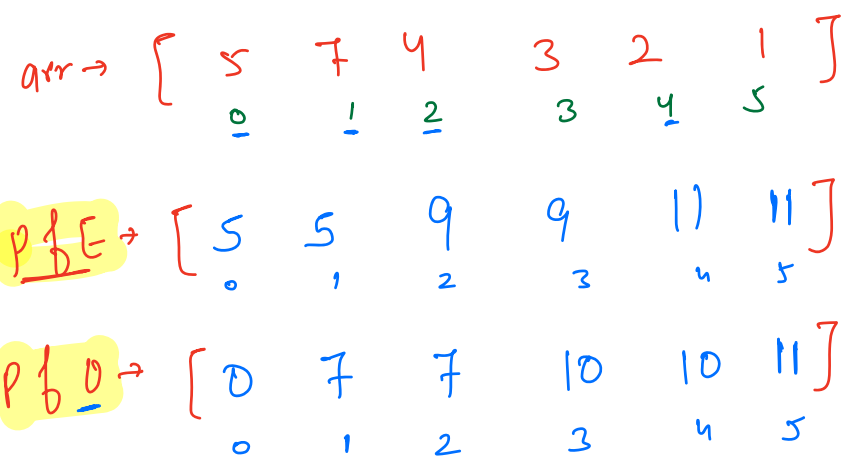
Today's Quote →

Content:

↳ 2 Google Problems



Q1) Special Index → {Hard}

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

Sum of all even index elements = Sum of all odd index elements.

Count total no. of special index.

Eg: arr[6] : [4 (0) 3 (1) 2 (2) 7 (3) 6 (4) -2 (5)] ans = 2

delete index - 0

cp → [3 (0) 2 (1) 7 (2) 6 (3) -2 (4)]

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

$S_o = 2+6 = 8$

delete index - 1

cp → [4 (0) 2 (1) 7 (2) 6 (3) -2 (4)]

$S_e = 9$, $S_o = 8$

delete index - 2

cp → [4 (0) 3 (1) 7 (2) 6 (3) -2 (4)]

$S_e = 9$, $S_o = 9$

delete index - 3

cp → [4 (0) 3 (1) 2 (2) 6 (3) -2 (4)]

$S_e = 4$, $S_o = 9$

delete index - 4

cp → [4 (0) 3 (1) 2 (2) 7 (3) -2 (4)]

$S_e = 4$, $S_o = 10$

delete index - 5

cp → [4 (0) 3 (1) 2 (2) 7 (3) 6 (4)]

$S_e = 12$, $S_o = 10$

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

pseudo-code.

```
int  specialIndices ( arr, N) {
    count = 0
    for ( i → 0 to N-1) {   → N iterations
        // Create cp [N-1] where arr[i] is removed
        // arr[N] = [0  1  2  3  --- i-1  i  i+1 -- N-1]      → N iterations
        // cp [N-1] = [   copy these        copy these]
        iterate and find Se , So .                          ] N iterations.
        if ( Se == So ) { count = count+1 }
    }
    return count
}
```

{Todo} ← (for the create/copy step)

$T.C = O(N^2)$
$S.C = O(N)$

idea 2 :

arr[10] : [3 2 6 8 | 9 | 9 7 6 4 12]
0 1 2 3 4 5 6 7 8 9

Delete index 4

odd → even
even → odd

Cp → [3 2 6 8 | 9 7 6 4 12]
0 1 2 3 4 5 6 7 8

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

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

Delete 5th index

even → odd
odd → even

Cp → [2 1 3 0 6 | 3 4 5 6 10 2]
0 1 2 3 4 5 6 7 8 9 10

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

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

$Cp S_o = Cp S_e[0-10] = Cp S_o[0-4] + Cp S_o[5-10]$

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

Generalisation

arr[N] : $[a_0\ (0)\ \ a_1\ (1)\ \ a_2\ (2)\ \ a_3\ (3)\ \ \cdots\ \ a_{i-1}\ (i-1)\ \ a_i\ (i)\ \ a_{i+1}\ (i+1)\ \ \cdots\ \ a_{N-2}\ (N-2)\ \ a_{N-1}\ (N-1)]$

Deleting i^{th} index

cp → $[a_0\ (0)\ \ a_1\ (1)\ \ a_2\ (2)\ \ a_3\ (3)\ \ \cdots\ \ a_{i-1}\ (i-1)\ \ a_{i+1}\ (i)\ \ \cdots\ \ a_{N-2}\ (N-3)\ \ a_{N-1}\ (N-2)]$

$S_e = cp\,S_e[0 \to N-2] = cp\,S_e[0 \to i-1] + cp\,S_e[i \to N-2]$

S_e after deleting i^{th} index = $arr\,S_e[0 \to i-1] + arr\,S_o[i+1 \to N-1]$

S_o after deleting i^{th} index = $arr\,S_o[0 \to i-1] + arr\,S_e[i+1 \to N-1]$

$Sum(i, j) = pSum[j] - pSum[i-1]$.

Final observations

$S_e = S_e[0 \to i-1] + S_o[i+1, N-1]$

$\{ S_e = pfE[i-1] + pfO[N-1] - pfO[i] \}$

$i = 0$

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

$\{ S_o = pfO[i-1] + pfE[N-1] - pfE[i] \}$

$i = 0$

pseudo code →

```
int specialIndices(arr, N) {
    // pfE[N], pfO[N]
    count = 0
    for(i → 0 to N-1) {
        // we are going to delete ith index.
        Se = pfO[N-1] - pfO[i]
        if(i != 0) { Se = Se + pfE[i-1] }
        So = pfE[N-1] - pfE[i]
        if(i != 0) { So = So + pfO[i-1] }
        if(Se == So) { count += 1 }
    }
    return count
}
```

→ 2N iterations, 2N space.

→ N

T.C → O(N)
S.C → O(N)

Break till 10:42

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

An element with freq > N/2.

Eg1: arr[6] : { 1 2 1 6 1 1 } 4 > 6/2 , ans = 1

(0 1 2 3 4 5)

Eg2 : arr[9] : { 3 4 4 8 4 9 4 3 4 } 5 > 9/2

(0 1 2 3 4 5 6 7 8)

ans = 4

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

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

3 → 7 times 7 > 12/2 ans → 3

Eg4: arr[10] : { 4 3 3 3 4 3 4 4 4 3 }

(0 1 2 3 4 5 6 7 8 9)

4 → 5 > 10/2

3 → 5 > 10/2

{There is no majority element}

Ideas:

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

Nested loops will be required. T.C → O(N²)

② Sort the array [N log N] T.C → O(N log N)

arr → { 5 3 1 5 7 5 5 }

arr → { 1 3 5 5 5 5 7 } [N]

Observations : arr[N].

At max how many majority elements can we have?

// Let's say we have 2 majority elements. → m1, → m2

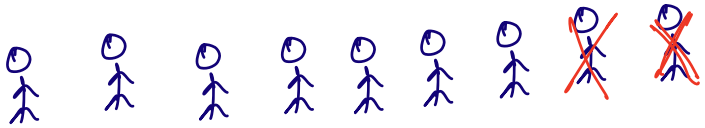
$$freq(m1) > N/2$$

$$freq(m2) > N/2$$

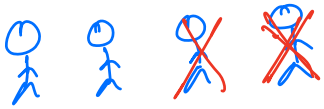
$$freq(m1) + freq(m2) > N$$

∴ By contradiction, there can be at max 1 majority element.

Elections : 15 MLA.

Mayank → 

Aravindhan → 

Geetanjali → 

Mayank : 9 > 15/2 [Yes].

Mayank : 8 > 13/2 [Yes]

Mayank : 7 > 11/2 [Yes]

Mayank : 7 > 9/2 [Yes].

Obs 1 :

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

Mayank : 

Aravindhan : 

Geetanjali : 

Obs2: If we remove 2 identical items, then majority can change.

Eg: arr[9]: { 4 4 3 8 8 4 9 4 4 }
0 1 2 3 4 5 6 7 8

freq(4) > 9/2 : Yes.

freq(4) > 7/2 : Yes.

freq(4) > 5/2 : Yes.

freq(4) > 3/2 : Yes.

[How can we delete two distinct elements always?]

arr[9]: { 4 4 3 8 8 4 9 4 4 10 }
0 1 2 3 4 5 6 7 8 9

element = 4 8 9 4

freq = 2 1 0 1 0 1 0 1 0

arr[10]: { 4 2 5 2 7 4 4 }
0 1 2 3 4 5 6

element = 4 8 7 4

freq = 1 0 1 0 1 0 1

[No majority element]

pseudo code:

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

Moore's Voting Algorithm

T.C → O(N)

S.C → O(1)

Doubts →

arr → [2 1 5 5 4 2 2 2 2]
0 1 2 3 4 5 6 7 8

element = 2 5 2

freq = 1 0 1 2 1 0 1 2 3

arr → [1 2 1 2 1 2 3 2 5 2]
0 1 2 3 4 5 6 7 8 9

element = 1 2 1 5 5

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

[Majority element:
⇓
element whose frequency > N/3]

N/m

⇓

[Delete 3 distinct items.] → P.S.S.

→ Try to code every approach that you can think of.
try to take help → [peers, TA's, me.]