Lecture :- Interview problems 1

Agenda

- Special Index ✓
- Majority element 1. ✓
- Majority element 2.

class starts at 7:05 AM

Qu1 Special index

An idx is said to be special index, if after deleting that idx —

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

$$arr[\,] = \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5. \\ 4 & 3 & 2 & 7 & 6 & -2 \end{array}\right] \Rightarrow$$ 2 special idx.

idx	updated array	S_e	S_o	special idx
0	0 1 2 3 4 3 2 7 6 -2	3 + 7 −2 = 8	2 + 6 = 8	Yes
1	0 1 2 3 4 4 2 7 6 -2	4 + 7 -2 = 9	2 + 6 = 8	NO
2	0 1 2 3 4 4 3 7 6 -2	4 + 7−2 = 9	3 + 6 = 9	Yes
⋮	⋮	⋮	⋮	⋮
5	0 1 2 3 4 4 3 2 7 6	4 + 2 + 6 = 12	3 + 7 = 10	NO

Brute force:

```
└ Iterate from 0 to n-1 indices —
      Create a new array [ ith idx won't exist]
      Calculate Se and So
      if (Se == So)     special idx
      else              do nothing
```

Observations

	0	1	2	3	4	5	6	7	8	9	10	
arr[] = [	2	3	8	7	4	5	1	2	6	9	5	]

Delete the 4th idx —

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

1. (0, 3) idx remain unchanged.

2. 5th idx becomes 4th idx

   6th idx becomes 5th idx

   7th idx becomes 6th idx.

   ⋮ ⋮ ⋮ ⋮

Odd indices el becomes even indices el

Even " " " odd " ".

arr[] = [

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

]

case1: Delete 3rd idx —

arr[] = [

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

]

S_e = sum of even[0, 2] + sum of odd[4, 8]

4 + 2   -2 + -1

S_o = sum of odd[0,2] + sum of even[4, 8]

3   6 + 9 + 3.

arr[] = [

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

]

Delete 4th idx —

arr[] = [

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

]

S_e = sum of even[0, 3] + sum of odd[5, 8]

S_o = sum of odd[0,3] + sum of even[5, 8]

⋮   ⋮   ⋮

Generalization:

Delete ith idx —

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

$= pfEven[i-1] + pfodd[n-1] - pfxodd[i+\cancel{1}-\cancel{1}]$

$S_e = pfEven[i-1] + pfodd[n-1] - pfxodd[i]$.

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

$S_o = pfodd[i-1] + pfEven[n-1] - pfEven[i]$

Edge case:

$i == 0$

idx out of bound exception

$S_e = pfEven[0-1]$

$S_e = pfodd[n-1] - pfodd[i]$

$S_o = pfEven[n-1] - pfEven[i]$.

```
int countSpecialIndices(int[] arr) {

        int n = arr.length;

TC: O(n)  ←  int[] pfEven = prefixEvenOptimal(arr); → sc: O(n)

TC: O(n)  ←  int[] pfOdd = prefixOddOptimal(arr);  : sc: O(n).

        int count = 0;

TC: O(n)  ←  for( i=0; i<n; i++) {

            if (i==0) {

                Se = pfodd[n-1] - pfodd[i];

                So = pfEven[n-1] - pfEven[i];

            } else {
                Se = pfEven[i-1] + pfodd[n-1] - pfodd[i];

                So = pfodd[i-1] + pfEven[n-1] - pfEven[i];
        }
```

```
            if ( se == so) {
                  count++;
            }
        }
        return count;
    }
```

TC: $O(n)$

SC: $O(n)$.

Assignment: Can we do this in $O(1)$ space? [carry forward]

Qu2 Majority el.

Given an arr, find majority el.

Majority el: An el whose freq > $\frac{n}{2}$.

Note: Majority el will always be present in array.

arr[6] = [1 2 1 6 1 1] → ans = 1 [freq = 4 > $\frac{6}{2}$]

arr[9] = [3 4 4 8 4 9 4 3 4] → ans = 4

arr[10] = [4 6 5 3 4 5 4 4 4 8] → invalid if

Majority el = 4 [~~freq = 5 >~~ $\frac{10}{2}$]

Possible approaches

1. Nested loop → $O(n^2)$.

2. Sort the array. → $O(n\log n)$.

arr[9] = [3 4 4 8 4 9 4 3 4]

↓ sort

arr[9] = [3 3 4 4 4 4 4 8 9]

3. Hashmap [TC: O(n) SC: O(n)]

Expectation:- TC: O(n)
SC: O(1)

Observations 1

Q How many majority el can be there in an array?

Ans: Max of 1 majority el.

arr[n] = [_ , _ , _ , _ , _ , _ , _ , . . . _ .]

majority el (freq) = $\frac{n}{2} + 1$ [min]

Remaining el = $\frac{n}{2} - 1$ [No el can have freq greater than $\frac{n}{2}$].

Observation 2

15 people in a village

Bala (Alcohol)

Shivani (Education)

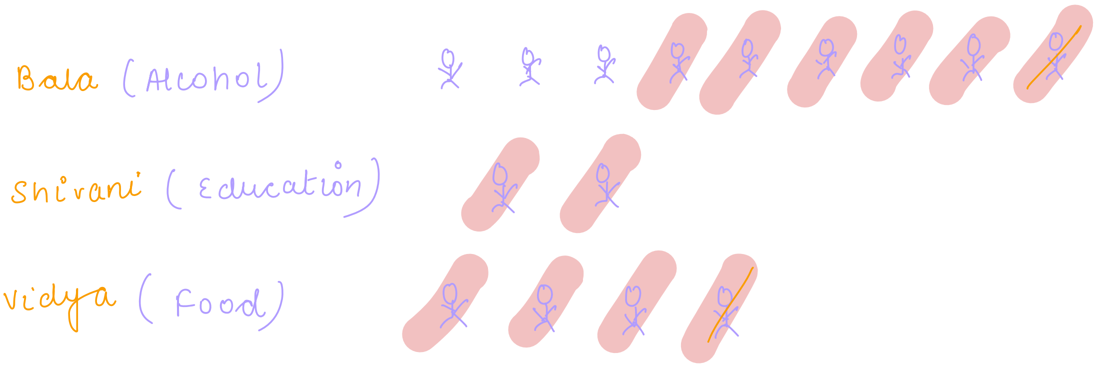
Vidya (Food)

Even if Shivani and Vidya do महागठबंधन ,

Bala is still winner [> $\frac{n}{2}$ votes]

→ Ankit is a criminal.

Bala hires Ankit. [kill 1 of my people
kill 1 of shivani | vidya people]



This will ensure, that at least one vote of Bala is left after all the people of shivani and vidya are killed by Ankit.

Establish an observation

arr[9] = [4 4 3 8 8 4 9 4 4]

majority = 4 4 4 4 4

Non-majority = 3 8 8 9

cancel | kill 2 el :- 1 el from majority &
1 el from non-majority
You will be left with majority el

Break: 8: 40 AM

final observation

majority el and non-majority el are always distinct.

arr[9] = [4 4 3 8 8 4 9 4 4]

majority el: 4 4 4 4 4

non-majority: 3 8 8 9

case 1: cancel one majority and one non-majority (Distinct)

majority el: 4 4 4 4 4

non-majority: 3 8 8 9

→ left with 4

case 2: cancel non-majority el.

majority el: 4 4 4 4 4

non-majority: 3 8 8 9

→ left with 4.

case 3: cancel two majority el. [Non- distinct)

majority el: 4 4 4 4 4

Moore Voting Algorithm

$$arr[11] = \left[\begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 3 & 4 & 6 & 1 & 3 & 2 & 5 & 3 & 3 & 3 \end{array} \right]$$

candidate MajorityEl = 3

freq = 1.

i = 1	arr[1] = 3 , majority = 3.
	arr[1] & majority el are non- district
	DO not kill
	majority = 3. freq = 2

i = 2	arr[2] = 4 , majority = 3
	arr[2] & majority el are distinct.
	kill both of them
	majority = 3 freq = 1

i = 3	arr[3] = 6 , majority = 3
	arr[3] & majority are distinct
	kill them
	majority = 3. , freq = 0 [Not possible)
	majority = none

$arr[11] = \left[\begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 3 & 4 & 6 & 1 & 3 & 2 & 5 & 3 & 3 & 3 \end{array}\right]$

i=4	arr[4] =1 , majority = none.
	majority =1 , freq =1.

i=5	arr[5] =3 , majority =1.
	Distinct
	kill them
	majority = none , freq =0

i=6	arr[6] = 2 , majority = none
	majority = 2 , freq =1.

i=7.	arr[7] =5 , majority = 2.
	Distinct
	kill them
	majority = none , freq =0

i=8	arr[8] = 3, majority =none
	majority =3 , freq =1

i=9	arr[9] =3, majority =3
	majority =3 freq =2

i=10	arr[10] =3, majority =3
	majority =3 freq =3

H|W:- What if there are no majority el in array?

TC: O(n)

SC: O(1)

[freq = 0] → check ?

```
int majorityEl(int[] arr) {
    int majority = arr[0];
    int freq = 1;

    for( i=1; i<arr.length; i++) {
        if ( freq==0) {
            majority = arr[i];
            freq=1;
        } else if( arr[i] == majority) {
            freq++;
        } else {   // Distinct el.
            freq--;
        }
    }
    return majority;
}
```

TC: O(n)

SC: O(1)

Qu Majority el 2.

Given an arr, find majority el.

Majority el: An el whose freq $> \frac{n}{3}$.

Note: Majority el will always be present in array.

Observation: How many majority el are possible?

2 max.

arr[n] = [__ , __ , __ , ...]

freq(majority 1) = $\frac{n}{3} + 1$.

freq(majority 2) = $\frac{n}{3} + 1$.

$$\left\{ \frac{n}{3}+1 + \frac{n}{3}+1 = \frac{2n}{3} + 2 = \frac{2n+6}{3} \right\}$$

$$\frac{2n+6}{3} < n.$$

freq ✗ (majority 3) = $\frac{n}{3} + 1$.

$$\left\{ \frac{n}{3}+1 + \frac{n}{3}+1 + \frac{n}{3}+1 = n + 3. \right\}$$

Basic idea

majority el1 = arr[0]   majority el2 = arr[0] || arr[1]

freq1 = 1   freq2 = 1

arr[i] == majority el1 → freq1++

else arr[i] == majority el2 → freq2++

else   freq1--
          freq2--

Thankyou ☺

Contest :- friday 7:00 - 8:30 AM.

2 Que must out of 3.

Post 8:30 — contest sol discussions.

Topics: Arrays
- prefix sum
- carry forward

Doubt

arr[100] = [_ , _ , _ , _ , _ , _ , _ , _]

$freq(majority1) \ngtr \frac{n}{3} = \frac{n}{3} + 1 \Rightarrow \boxed{34}$

Remaining el = 66.

freq(majority2) = 34.

Remaing el = $\boxed{32.}$ ✓

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ arr[7] = [& \cancel{2} & 4 & 8 & 1 & 2 & 6 & 2 \;] \\ pfodd[] & 0 & 4 & 4 & 12 & 12 & 18 & 18 \end{array}$$

0 th idx deleted

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ arr[7] = [& 4 & 8 & 1 & 2 & 6 & 2 \;] \end{array}$$

pfodd() = [0 8 8 10 10 12

pfEven() [

$pfodd(6) - \boxed{pfodd[0]} = 0$ ✓

$S_e = S_o[1 - 6] \rightarrow$

$S_o = S_e[1 - 6]$

$pfodd[l, r] = pfodd(r) - pfodd(l-1)$