Arrays : Interview Problems

Jul 12, 2023

AGENDA

- Sum of even-indexed elements
- Special index
- Majority element

Q (Directi) Given N array elements, and Q queries (st, end), for every query → return the sum of all even-indexed elements from st to end.

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

st	end	sum
1	3	1
2	5	5
0	4	7
3	3	0

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

Sum of even-indexed elem from i to j

= Sum of even-indexed elem from 0 to j

− Sum of " " from 0 to $i-1$

	0	1	2	3	4	5
	2	3	1	6	4	5
prefix-sum :	2	2	3	3	7	7

$$\begin{tikzcd}\boxed{N*Q} \arrow[d] \\ \boxed{N+Q}\end{tikzcd}$$

	0	1	2	3	4	
	2	4	3	1	5	
prefix	2	6	9	10	15	x x

10-2=8

pf[i] ⇒ sum of elements from 0 to i.

Requir: pf[i] ⇒ sum of even-indexed elements from 0 to i.

	0	1	2	3	4
	2	4	3	1	5
Prefix	2	2	5	5	10

pf[i] → sum of even from 0 to i.

$$pf[i] \rightarrow \begin{cases} pf[i-1] & \text{if } i \text{ is odd} \\ pf[i-1] + arr[i] & \text{if } i \text{ is even} \end{cases}$$

Code

```
// Create prefix array
pf[0] = arr[0]
for(int i=1; i<n; i++)
{
        if (i%2 == 0)
        {
            pf[i] = pf[i-1] + arr[i]
        } else
            pf[i] = pf[i-1]
```

0	1	2	3	4
2	4	3	1	5

2 → 2 ↗ 5 → 5 ↗ 10

```
}

// Answer queries.
for(int i=0; i<Q ;i++)
{
            st. = LEFT[i]
            end = RIGHT[i]
             sum = pf[end] - pf[st-1];
                              ↳ Handle edge case
                                  if st==0.
                                    ↓
                                 sum=pf[end];

 }
```

T.C. → O(N+Q)
S.C. → O(N)

Sum of odd-indexed elements

	0	1	2	3	4	5
	2,	3,	1,	6,	4,	5
prefix:	0	3	3	9	9	14

↘ Pf[i] → sum of odd-indexed element from 0 to i.

Code →

```
// Create prefix array
pf[0] = 0
for(int i=1; i<n;i++)
{
        if (i%2 != 0)
        {
              pf[i] = pf[i-1] + arr[i]
        } else
              pf[i] = pf[i-1]
}
```

Q.

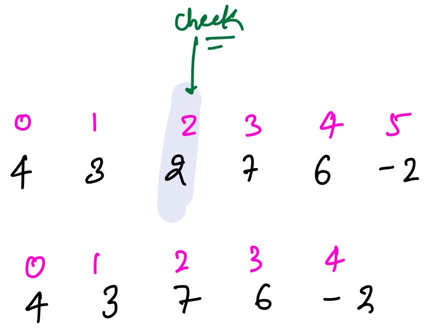
Facebook
Google
Directi

Given N array elements, count no. of special index in the array.

Special index → are those after removing which, sum of odd-indexed element becomes equal to sum of even-indexed elements.

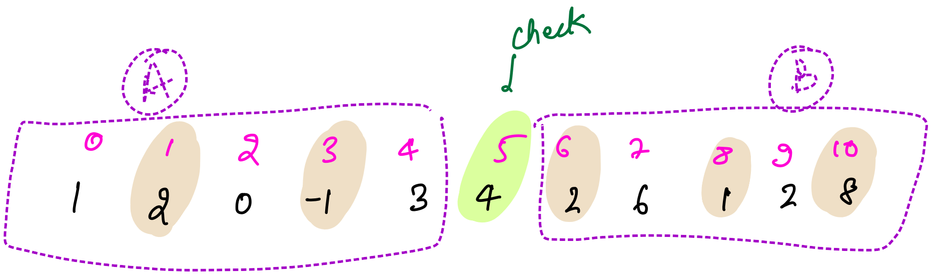
0	1	2	3	4	5
4	3	2	7	6	-2

index	New-array	Se	So	
0	3 2 7 6 -2	8	8	✓
1	4 2 7 6 -2	9	8	✗
2	4 3 7 6 -2	9	9	✓
⋮				



On removing i,

elements after i → indexes are reversed.
odd becomes even.
even becomes odd.



Sum of odd-indexed elements after removing i

= sum of odd-indexed elements from 0 to i-1. +
Sum of even-indexed elements from i+1 to n-1.

Sum of even-indexed elements after removing i

= Sum of even-indexed from 0 to i-1 +
Sum of odd-indexed from i+1 to n-1

				l ↓		r ↓
	0	1	2	3	4	5
	4	3	2	7	6	-2
pfEven →	4	4	6	6	12	12
pfOdd →	0	3	3	10	10	8

check if i=2 is special.

Sum of even-indexed elements after removing i

$$= \text{SumEven from } \underline{\overset{l}{0}, \overset{r}{i-1}} + \text{SumOdd from } \underline{\overset{l}{i+1}, \overset{r}{n-1}}$$

$$= \underbrace{pfEven[i-1]}_{pf[r]} + \underbrace{pfOdd[n-1]}_{pf[r]} - \underbrace{pfOdd[i]}_{pf[l-1]}$$

Sum of odd-indexed elements after removing i

$$= \text{SumOdd from } 0 \text{ to } i-1 + \text{SumEven from } i+1 \text{ to } n-1$$

$$= pfOdd[i-1] + pfEven[n-1] - pfEven[i]$$

Code.

```
//Pre-processing.
int pfEven[]
int pfodd[]
pfEven[0] = arr[0]
pfodd[0] = 0
for(int i=1; i<n; i++)
{
        if (i%2==0)
        {
                pfEven[i]= pfEven[i-1] +arr[i]
                pfodd[i]= pfodd[i-1]

        }else
        {
                pfEven[i] = pfEven[i-1]
                pfodd[i] = pfodd[i-1]+ arr[i]

        }

}

ans=0
for(int i=0; i<n; i++)
{
                                                  if i==0, this is 0.
        //Check if i is special.
        sumOdd =   pfOdd[i-1] +   pfEven[n-1] - pfEven[i]
                     Bucket A          Bucket B

        sumEven =  pfEven[i-1] +  pfodd[n-1] - pfOdd[i]
```

```
            if (sumodd == sumEven)
                  ans++;

      }
      return ans;
  }
```

T.C. → O(N)

S.C. → O(N)

Break till 8:25 AM

(Google)

Q. Given N array elements, find the majority element. If it doesn't exist, return −1.

A majority elem → which occurs more than N/2 times in the array.

{2, 1, 1, 1, 3, 4, 6} × No majority elem.

{3 4 3 2 4 4 4} → 4 ✓

3 4 3 6 1 3 2 5 3 3 3 → $\boxed{3}$ ✓ $\boxed{6}$

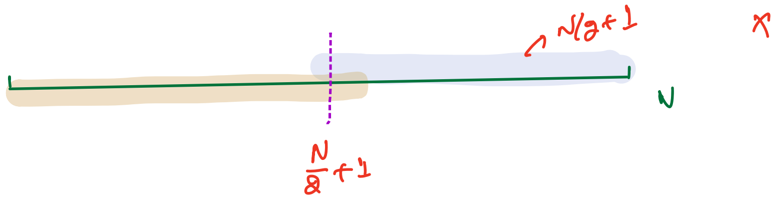
$\{\underbrace{4\ 3\ 1\ 6\ 4\ 4\ 4\ 3}_{\text{No majority.}}\}$

8 elements
↓
5 elements.

= doesn't work.

* How many majority elements in the array?

At max - 1.



3 4 3 2 4 4 4

→ Count frequency of each element if any frequency > N/2, it is majority.

T.C. → O(N)

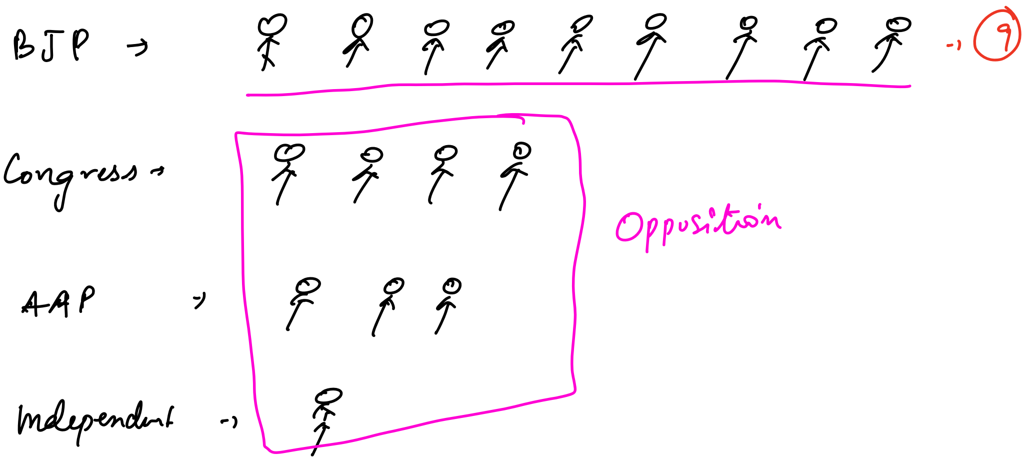
S.C. → O(N) ← HashMap

xx

xx

Moore's voting algorithm

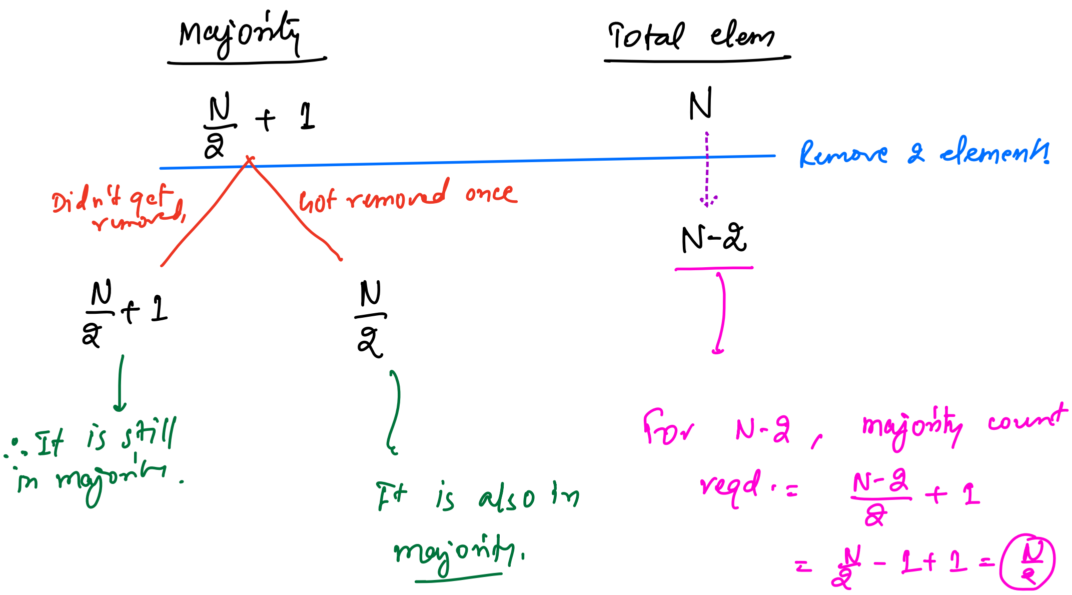
A



** Remove 2 people from different parties.

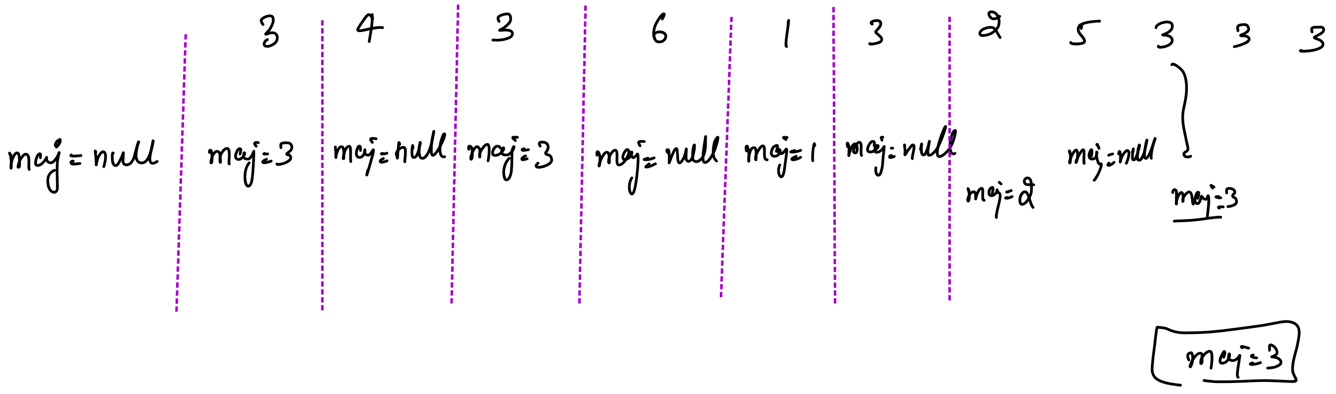
** If 2 distinct nos. from the array is removed, majority element doesn't change.

3 4 2 3 4 4 4

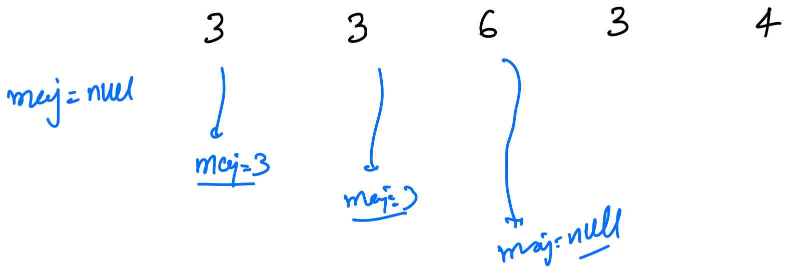


For N-2, majority count reqd. = $\frac{N-2}{2}+1$

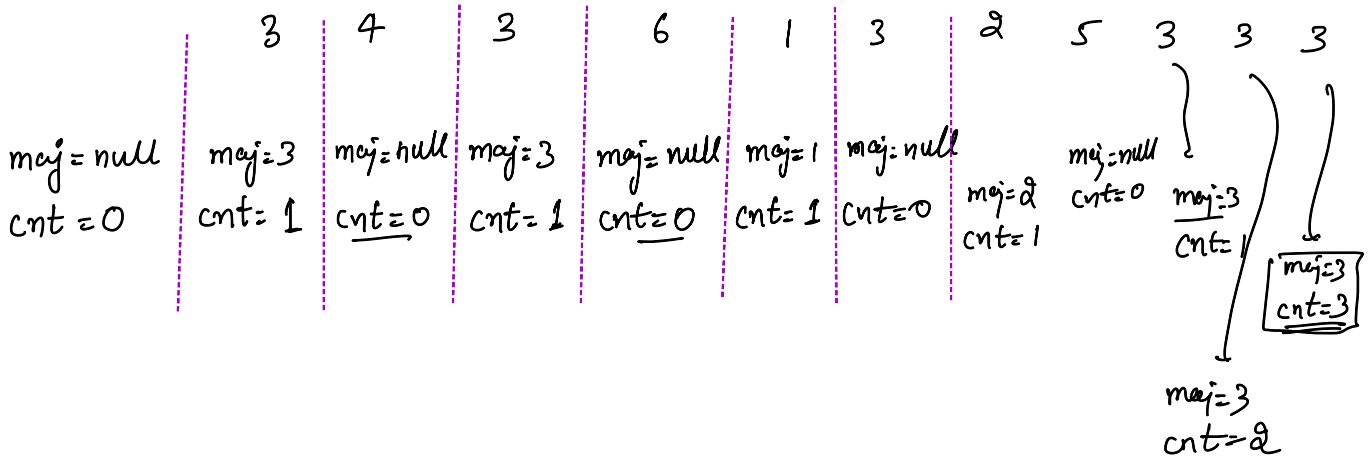
$= \frac{N}{2} - 1 + 1 = \boxed{\frac{N}{2}}$



3 4 3 6 1 3 2 5 3 3 3
maj = null
maj=3
maj=null
maj=3
maj=null
maj=1
maj=null
maj=2
maj=null
maj=3
maj=3



3 3 6 3 4
maj=null
maj=3
maj=3
maj=null



3 4 3 6 1 3 2 5 3 3 3
maj=null
cnt =0
maj=3
cnt= 1
maj=null
cnt=0
maj=3
cnt= 1
maj= null
cnt=0
maj=1
cnt= 1
maj=null
cnt=0
maj=2
cnt=1
maj=null
cnt=0
maj=3
cnt=1
maj=3
cnt=2
maj=3
cnt=3

2	2	3	2	3	5	3	4	3	3	3	4
m=2	m=2	m=2	m=2	m=2	m=null	m=3	m=null	m=3	m=3	m=3	m=3
c=1	c=2	c=1	c=2	c=1	c=0	c=1	c=0	c=1	c=2	c=3	c=2

Majority = 3

1	2	3	4	1	1
m=1	m=null	m=3	m=null	m=1	m=1
c=1	c=0	c=1	c=0	c=1	c=2

1 is in majority ?

Moore's voting algo → doesn't give you the majority element.
it gives you the candidate of majority.

1	2	1	2	1	2	3
m=1	m=null	m=1	m=null	m=1	m=null	m=3
c=1	c=0	c=1	c=0	c=1	c=0	c=1

* To handle this, do a double-check.
find the count of candidate and check
if it is > N/2.

0	1	2	3	4	5	6
1	1	2	2	1	1	2
m=1 c=1	m=1 c=2	m=1 c=1	m=null c=0	m=1 c=1	m=1 c=2	m=1 c=1

1 ✓

Code.

```
me = arr[0]
cnt = 1
for(int i= 1; i<n; i++)
{
        if (cnt==0)  ->  means, no majority at this moment.
        {
                me = arr[i]  }  Starting from scratch.
                cnt= 1       }

        } else if (me == arr [i])
        {
                cnt ++;

        } else
        {
                cnt--;

        }

}
```

```
// 'me' is a candidate for majority.
cnt=0
for(int i= 0;i<n;i++)
{       if (me == arr[i])
                cnt++;

}
if (cnt > N/2)
        return me
else
        return -1          ← // No majority element.
}
```

Count frequency of 'me'.

$T.C. \to O(N)$

$\boxed{S.C. \to O(1)}$

$\boxed{H.W.}$

Find N/3 majority element.

0 is even $\quad\quad$ $\underline{0 \% 2 = 0}$ ✓

$O(N) + O(N) = \underline{\underline{O(N)}}$

All the best for contest ☺