Q1 Given an array of size n. Given Q queries. For each query you are given 2 indexes s and e. For every query return the sum of all even indexed elements in the range s to e.

Ex1 :

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

Q=4

s	e		Sum
1	3	$\rightarrow$	1
2	5	$\rightarrow$	5
0	4	$\rightarrow$	7
3	3	$\rightarrow$	0

Pseudo Code :

```
for (int i=0 ; i<q ; i++) {
    int s = Q [i] [0];
    int e = Q [i] [1];
    int sum =0;
    for (int j = s ; j <= e ; j++) {
        if (j%2 == 0)
            sum += arr [j];
    }
    print(sum);
}
```

Q [q] [2];

TC : O(qn)

SC : O(1)

	0	1	2	3	4	5
Ex1 :	2	3	1	6	4	5
Pf :	2	2	3	3	7	7

How to Solve a Query:

→ s , e   sum = Pf[e] - Pf[s-1]

	0	1	2	3	4	5
Ex1 :	2	3	1	6	4	5
Pf :	2	2	3	3	7	7

s = 1 , e = 5   : Pf[5] - Pf[0]

$7 - 2 = \boxed{5}$

Pseudo Code !

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

//O(n)

```
for (int i=0 ; i<q ; i++) {
    int s = a[i][0];
    int e = a[i][1];
    int sum;
    if (s == 0)
        sum = pf[e];
    else
        sum = pf[e] - pf[s-1];

    print(sum);
}
```

//O(q)

TC : O(n+q)

SC : O(n)

Q2 Special Index [Google, Direct-i, Codenation]

It is an index after deleting it,

Sum of even indexed element = Sum of odd indexed element

Find the count of special indexes.

Ex1

$$\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 2 & 7 & 6 & -2 \end{array}$$

arr[] = 4 3 2 7 6 -2

1) Checking for index 0

$\Rightarrow$ delete value at index 0.

$$arr[] = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ 3 & 2 & 7 & 6 & -2 \end{array}$$

$Sum_{even} \Rightarrow 8$

$Sum_{odd} \Rightarrow 8$

SPECIAL INDEX!

2) Checking for index 1

$\Rightarrow$ delete value at index 1.

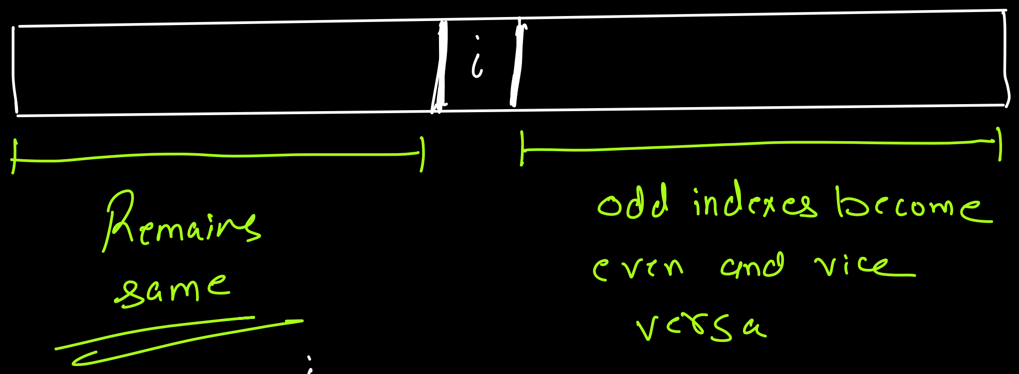
$$arr[] = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ 4 & 2 & 7 & 6 & -2 \end{array}$$

$Sum_{even} \Rightarrow 9$

$Sum_{odd} \Rightarrow 8$

NOT SPECIAL INDEX!!

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ arr[\,] = & 4 & 3 & 2 & 7 & 6 & -2 \end{array}$$



$$Pf_e[i] \Rightarrow \sum_{j=0}^{i} arr[j] \quad ; \; j \% 2 == 0$$

$$Pf_o[i] \Rightarrow \sum_{j=0}^{i} arr[j] \quad ; \; j \% 2 \; != 0$$

$$Sum_{even} = \left(\text{Sum of even in the first part} \right) + \left(\text{Sum of odd. in the second part} \right)$$

$$\left(Pf_e[i-1] \right) + \left(Pf_o[n-1] - Pf_o[i] \right)$$

$$Sum_{odd} = \left(Pf_o[i-1] \right) + \left(Pf_e[n-1] - Pf_e[i] \right)$$

	0	1	2	3	4	5
arr[] =	4	3	2	7	6	-2
Pf e =	4	4	6	6	12	12
Pf o =	0	3	3	10	10	8

1) index = 3

$Sum_{even} \Rightarrow Pf_e[i-1] + Pf_o[n-1] - Pf_o[i]$

$Pf_e[2] + Pf_o[5] - Pf_o[3]$

$6 + 8 - 10$

$\Rightarrow 14 - 10 = 4.$

$Sum_{odd} \Rightarrow Pf_o[i-1] + Pf_e[n-1] - Pf_e[i]$

$Pf_o[2] + Pf_e[5] - Pf_e[3]$

$3 + 12 - 6 \Rightarrow 9$

$Sum_{even} \neq Sum_{odd}$

Pseudo Code :

1) Calculate Pfeven [n]

2) Calculate Pfodd [n]:

```
    int cnt = 0;
    for (int i = 0 ; i < n ; i++) {
        int sum_e , sum_o;

        if (i = 0) {
            sum_e => Pfo[n-1] - Pfo[i];
        } else {
            sum_e => Pfe[i-1] + [ Pfo[n-1] - Pfo[i] ]
        }

        if (i == 0) {
            sum_o => Pfe[n-1] - Pfe[0];
        } else {
            sum_o => Pfo[i-1] + [ Pfe[n-1] - Pfe[i] ];

        if (sum_e == sum_o)
            cnt++;
    }
    return cnt;
```

TC : O(n)

SC : O(n)

Q Majority Element [Google, Facebook]

Given an array of +ve numbers. Return if there exists an element with frequency $> N/2$. [N is the length of arr]

SC: O(1).

Ex1 A:

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

freq > 3

$\Rightarrow$ 1

Ex2 A:

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

freq > 3

$\Rightarrow$ -1

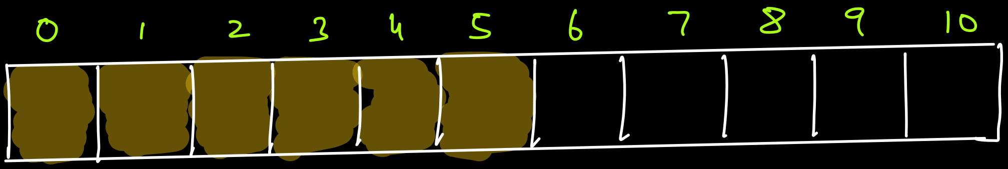
Brute Force

```
for (int i=0 ; i<n ; i++) {
    int val = arr[i];
    int cnt = 0;
    for (int j=0 ; j<n ;j++) {
        if (arr[j] == val)
            cnt ++;
    }
    if (cnt > (n/2))
        return val;
}
```

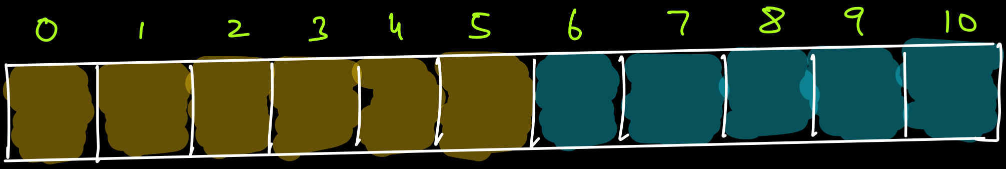
TC: $O(n^2)$

SC: $O(1)$

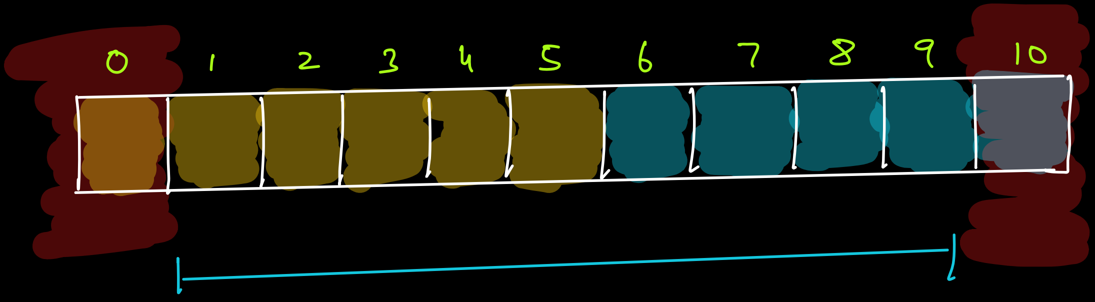
⇒ Only 1 majority element is possible.



CASE 1: Only 2 elements present

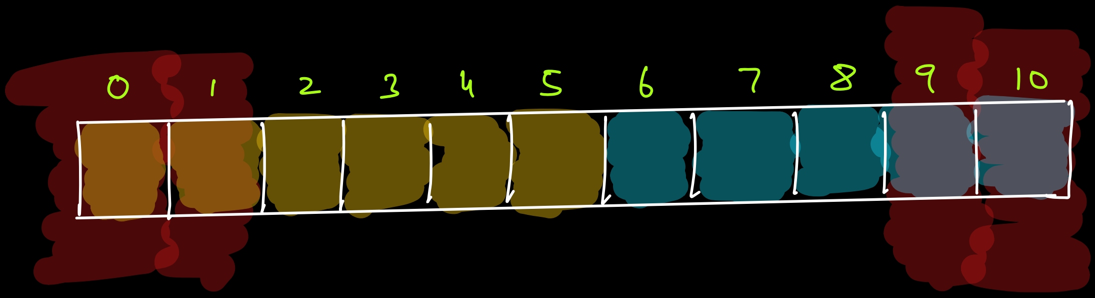


⇒ Remove 1 majority & 1 non majority.

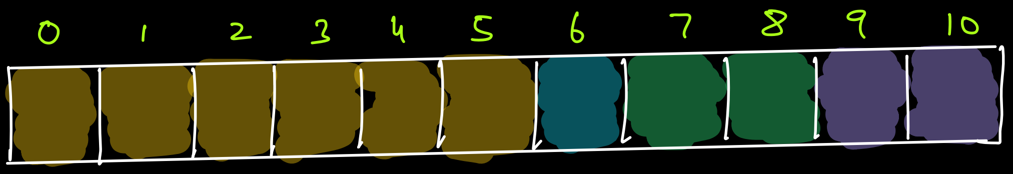


⇒ Majority remains same!

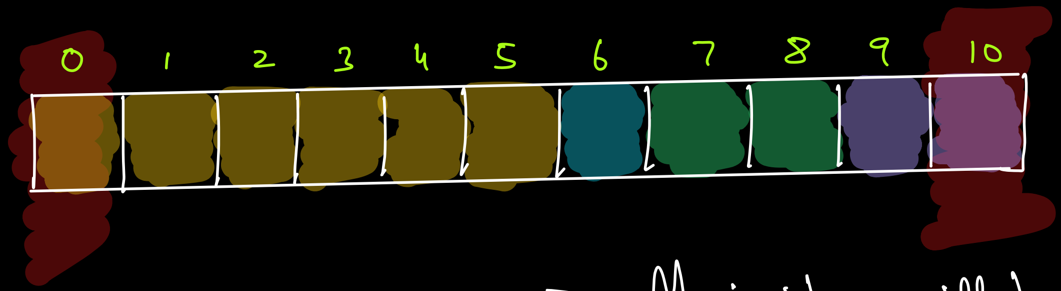
⇒ Remove 1 majority & 1 non majority.



CASE2: More than 2 elements present



⇒ Remove 1 majority & 1 non majority



⇒ Majority will be the same.

⇒ Remove 2 majority elements.

↳ Majority might change.

⇒ Remove 2 non majority elements.

⇒ Majority will be the same.

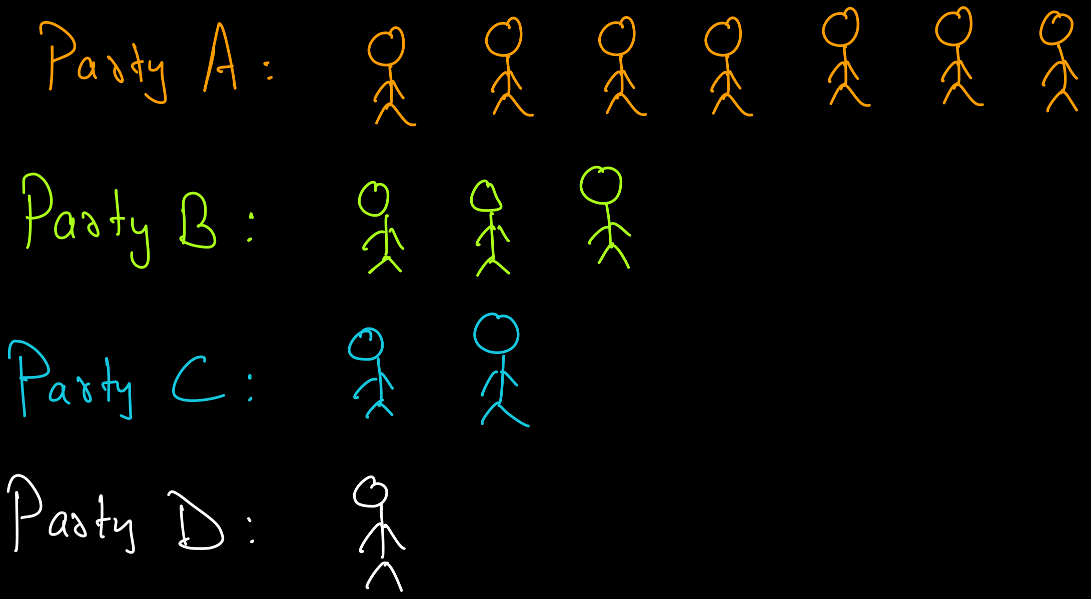
Removing 2 distinct elements doesn't change the majority!!

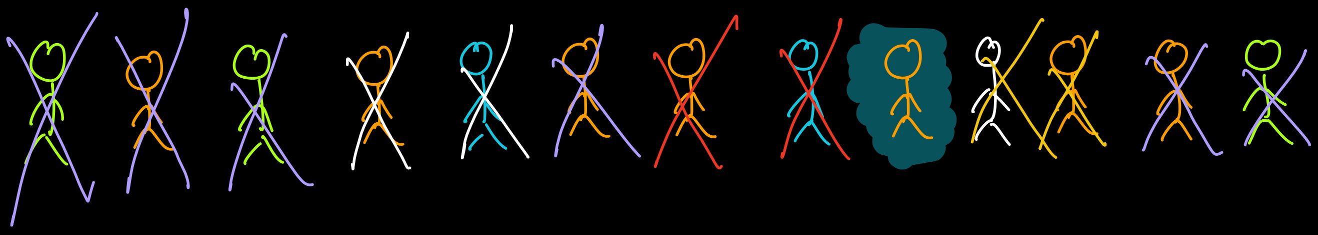
2 distinct elements : 1 maj & 1 non maj

: 2 non maj

Ex1 : $\cancel{3}$ $\cancel{4}$ $\cancel{3}$ $\cancel{6}$ $\cancel{1}$ $\cancel{3}$ $\cancel{2}$ $\cancel{5}$ $\cancel{3}$ $\cancel{2}$ 3

Ex2 : $\cancel{1}$ $\cancel{6}$ 1 $\cancel{1}$ $\cancel{2}$ 1

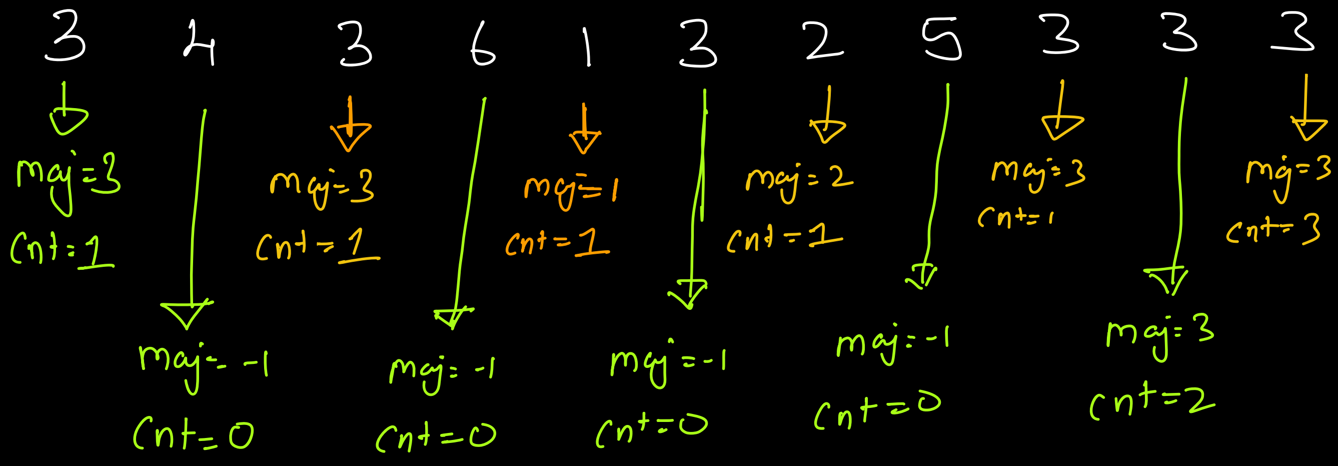




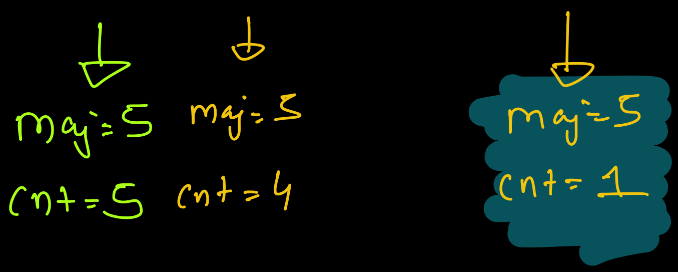
Ex1 : 3 4 3 6 1 3 2 5 3 3 3

$\Rightarrow$ maj = arr [0]

maj = 3 , cnt = 1



Ex2 5 5 5 5 5 3 3 3 3



Ex3

```
3   3   3 3   5   5   5   5   5
          ↓               ↓   ↳ maj = 5
        maj = 3       maj = -1  cnt = 1
        cnt = 4       cnt = 0
```

```
→ if new element == maj
          cnt ++;
  else new elent != maj
        if (cnt == 0)
            maj = newelement
            cnt = 1
      else
           cnt --;
```

Edge Case !

Ex

	1	2	1	1	3	4	5
maj	1	-1	1	1	1	-1	5
cnt	1	0	1	2	1	0	1

→ Loop once to count frequency of the majority element.