Today's Content :-

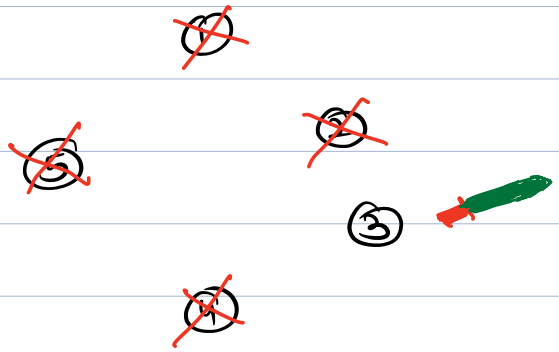
→ Josephus Problem

→ Majority Element

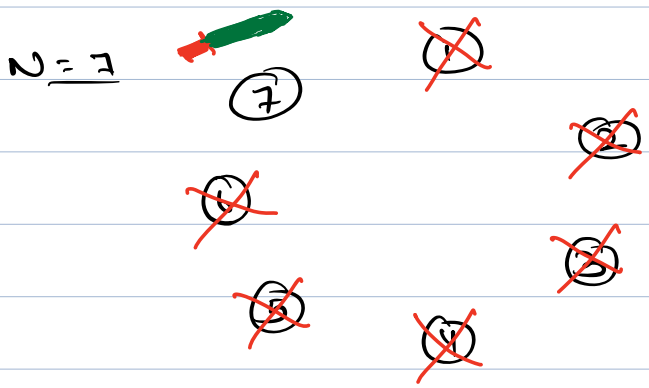
→ Doors

Ques) Josephus Problem :- Google / Adobe

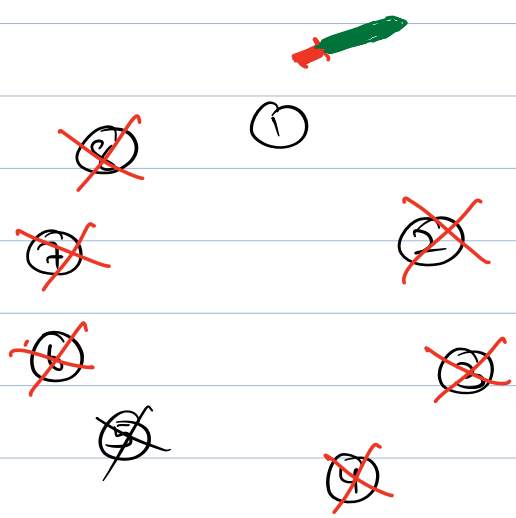
N = 5



N = 7



N = 8



N	ans
1	1
2	1
3	3
4	1
5	3
6	5
7	7
8	1
9	3
10	5
⋮	
16	1

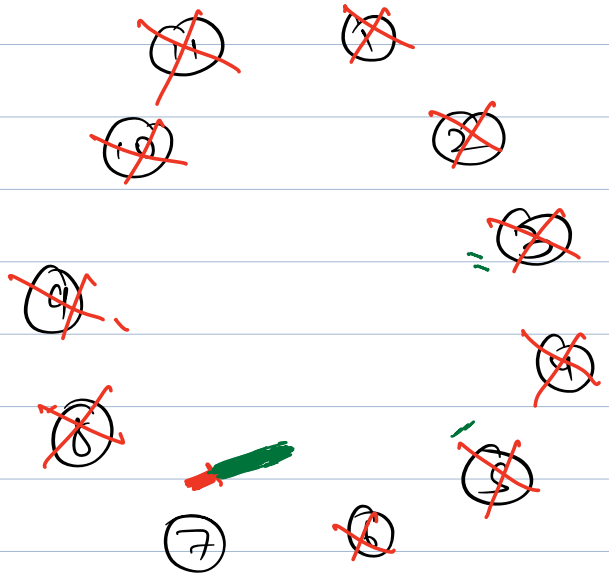
Observations :-

① Only odd no' can be ans.

② for no: 2^n, ans → 1.

N = 16

N = 11

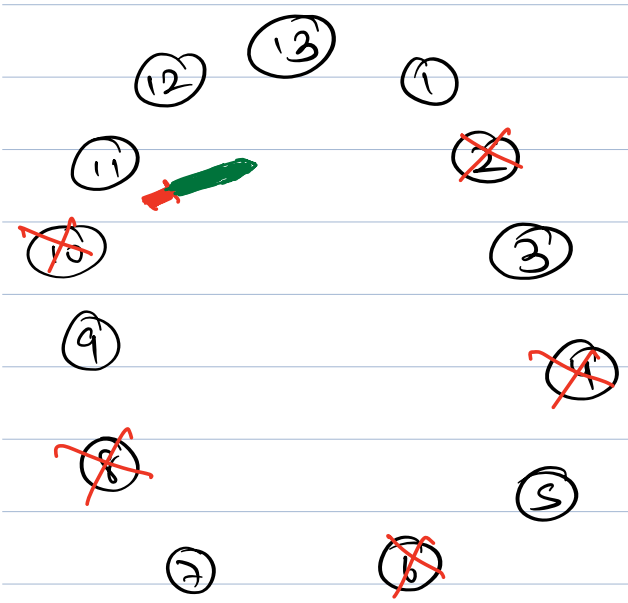


Remaining People	Starting Place
11	1
10	3
9	5
8	7

$13 - x = 8$

$x = 5$

N = 13



Remaining People	Starting Place
13	1
12	3 (+2)
11	5 (+2)
10	7 (+2)
9	9
8	11

1, 3, 5, 7, 9, 11 $\rightarrow (2n+1)$

$x - 5$ = nearest power of 2

$13 - 5$ = ____

$x - y \rightarrow 2y+1$.

1 → 3 → 5 → 7 ... $(2n+1)$

1 1 1 1

n

Steps :-

① find the nearest power of $2 \leq N$

$N - x = y$.

Total people → nearest power of 2. → y.

$\Rightarrow \quad x = N - y$.

② $2n+1$.

$\Rightarrow \quad N = 11, \quad y = 8.$

$x = 11 - 8 \Rightarrow 3.$

$\Rightarrow$ (7)

$\Rightarrow \quad N = 13, \quad y = 8.$

$13 - 8 = 5$

$\Rightarrow$ (11)

$N = 12$

$x = 17 - 16 \times 1$

$\Rightarrow 2n + 1 = 3.$

$N = 17.$ 5 mins

```
cp = 1
while (cp <= N)
{
    cp = cp * 2;
}
```

3

$\Rightarrow \frac{cp}{2}$

9:58 → 10:03 pm.

Ques) Majority Element Google MS, Adobe fb, GoldmanSachs.

Given an arr[], return if there exists a no. with frequency $> \frac{n}{2}$.

Eg 1) arr[6] = {1, 6, 1, 1, 2, 1}, $\frac{6}{2}$.

→ 1

Eg 2) arr[ ] = {3, 4, 3, 6, 1, 3, 2, 5, 3, 3, 3}

=> 3 $\frac{11}{2}$

Eg 3) arr[] = 4, 6, 5, 3, 4, 5, 6, 4, 4, 4 => $\frac{10}{2}$ => 4

:- Brute force :-

① → for all the element calculate freq.

T.C → O(n²)

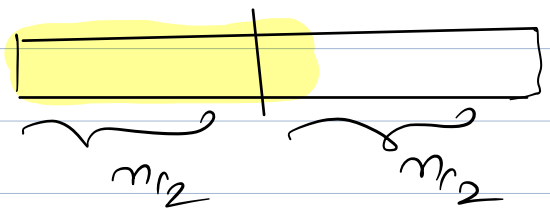
S.C → O(1)

② {1, 2, 3, 3, 3, 3, 3, 3, 4, 5, 6} Sorting

T.C → O(nlogn)

③ hashMap → T.C → O(N)
S.C → O(N)

Ques :- No. of majority Elements at most $\frac{n}{2}$.

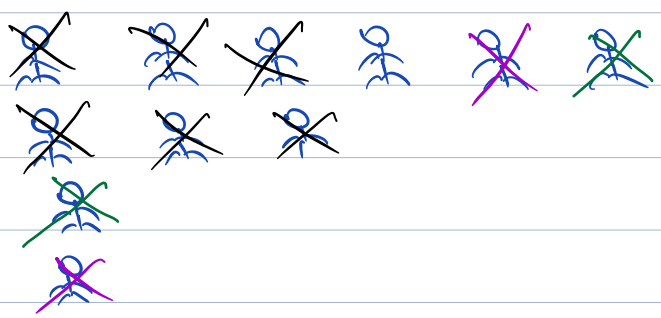


① Majority element will always be atmost 1.

② freq (majo) > freq (rest)

Run over even elections !!.

Aman
Sai
Priyanl
Vishal



Total = 11

Total	Majority
11	6
9	5
7	4
1	1

Conclusion:- If two distinct elements are removed, the majority still remains same.

0 1 2 3 4 5 6 7 8 9 10

3, 4, 3, 6, 1, 3, 2, 5, 3, 3, 3 N = 11.

majority => 6

maj = ~~3~~ ~~X~~ ~~3~~ ~~5~~ 3

freq = X 0 X 0 X 0 X 0 X 0 X 2

3 2 3 2 3 2 3 2 3 freq 5

maj = 3

freq = X 0 X 0 X 0 X 0 1

4 4 5 6 7 4 4 4 4 7 8

may = 4 freq = 6

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

4 4 6 6 8 7 7 7

may → 4 8 7

freq → 1 2 1 0 1 2 0

Aman
Priyanka
Mangesh
Rajat

T.C → O(N).
S.C → O(1).

Moore's Voting Algorithm

1, 1, 1, 2, 2, 2, 3

may = 3

freq = 1 2 3 2 1 0 1

Break :- 10:44 - 10:49

kumar.aman@scaler.com

Ques) Doors.

There are n doors,

Initially all are closed.

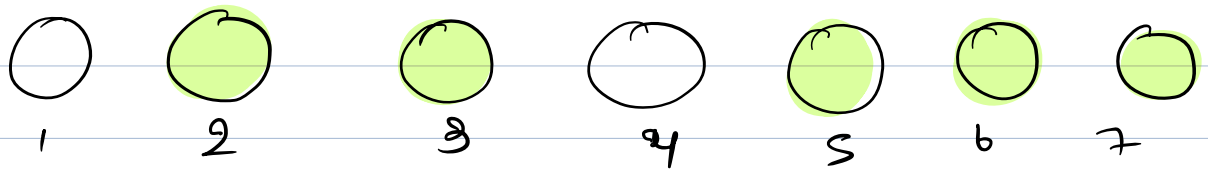
All the doors having 1 as their multiple will get toggled.

All the doors having 2 as their multiple will get toggled.

All the doors having 3 as their multiple will get toggled.

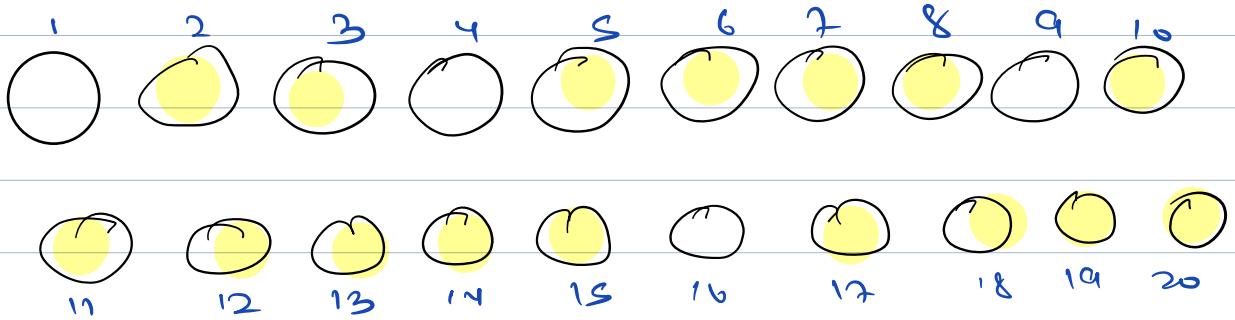
⋮

N = 7



Total doors opened → 2.

N = 20 Ans → 4



9 : 1, 3, 9 → open

15 : 1, 3, 5, 15 → close

18 : 1, 2, 3, 6, 9, 18 → close

if x has odd factors → open

if x has even factors → close.

25 : → 1, 5, 25

100

1	100
2	50
4	25
5	20
10	10

Perfect squares have odd factors.

(2∞) → stop it till $\sqrt{m}$.

for (i=1; i<=2∞; i++) {

1 → (1)
2 →
3
4
5
6
7
8
9
10
11

} Ans $\sqrt{m}$

T.C →

fun()

{

return $\sqrt{m}$

12
13
14
15

T.C → O(√n)

S.C → O(1)

ch = 'a' → 97

int x = ch - 'a'

x = 0

ch → 97

'a' → 97

int x = (int)ch

x = 97

ch = '0'

Str = '123' → 2

'1'

ch = Str.charAt(0)

ch

int n = ch - '0'

Sunday →

Saturday →

→ 1 + 2 + 3 + x + ... n = S1

− 1 + 2 + 3 + x + x + y ... n = S2

Str = '123'

ch = '1'

int = (int) ch

int = ch - '0'

1 - 1

'a' → 0
'b' → 1

ch - 'a'
↓
'b'
↓
98

char ch = 'c'
int n = ch - 'a'

c → 2

char ch = '1'
int n = ch - '0'

1, 2, 3 4, 44, , s, s

May → ✗ 3
Imp → ✗ 01

Majority Element Code :-

```java
public class Solution {
    // DO NOT MODIFY THE ARGUMENTS WITH "final" PREFIX. IT IS READ ONLY
    public int majorityElement(final int[] A) {
        int maj = A[0];
        int freq = 1;

        // Moore's voting Algorithm
        for(int i=1;i<A.length;i++){
            if(A[i]==maj)
            {
                freq++;

            }else if(freq==0){

                maj=A[i];
                freq=1;

            }else{
                freq--;

            }
        }


        return maj;

        // Here Question says Majority element will always be there,
        // so no need to check whether the final element left after above iteration has
        // frequency greater than n/2 or not.

    }
}
```