Arrays and Maths

22nd April, 2022

AGENDA:

- 3 very interesting problems.
  * N Doors problem
  * Nth Magical number
  * Majority Element
    ↳ (Moore's voting algo)

Q. N doors problem

Initially all doors are closed,

1st person comes and opens all doors.
2nd person comes, toggles all doors which are multiples of 2.

3rd person comes. toggles all " " multiples of 3.

⋮

ith per comes " " " multiples of i.

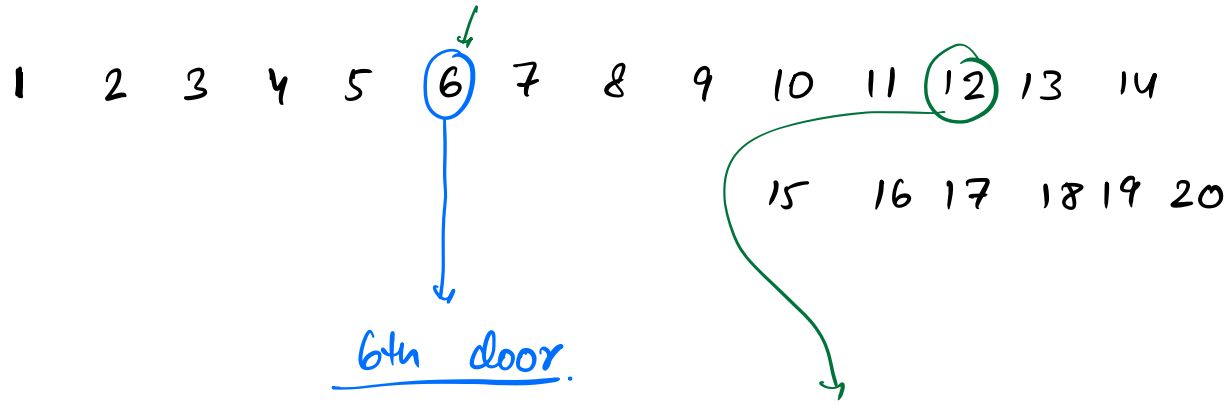
⋮

Nth person.

* Which doors remain open at the end of this experiment?

	1	2	3	4	5
Initially	×	×	×	×	×
1st.	✓	✓	✓	✓	✓
2nd	✓	×	✓	×	✓
3rd	✓	×	×	×	✓
4th	✓	×	×	✓	✓
5th.	✓	×	×	✓	×

1st , 4th door



6th door.

1, 2, 3, 6

1, 2, 3, 4, 6, 12.

Door was closed ⟶ Toggle 4 times → closed.

Door " ⟶ Toggle 7 times → open.

If door has been toggled (even) no. of times, it will remain closed.

" " (odd) no. of times, it will remain open.

** Doors which have odd no. of factors will remain open.

From 1 to N, which nos. have odd no. of factors ?

** factors always occur in pairs.

24

1,24 / 2,12 / 3,8 / 4,6

36 1,36 / 2,18 / 3,12 / 4,9 / 6,6

→ Odd no. of factors.

Perfect squares →

100.

1,100 / 2,50 / 4,25 / 5,20 / 10,10

9 factors

1, 2, 3, 4 . . . . . 17, 18, 19, 20

1, 4, 9, 16

N=200 $O(N^2)$

Pseudo-code

N

[0 0 0 0 0 0 0 0 0 0 0 0 . . . . 0]

for 1 to N : ← N person,

for ele in arr: ← Each door

$O(N^2)$

* Find all perfect squares from 1 to N.

```
ans = []
for i in range (1, N+1):
    if is_square(i):
        ans.append(i)
return ans.
```

O(N)

* Generate all perfect squares.

```
// N is input.
i = 1
while (i*i <= N):
    ans.append(i*i)
    i += 1
return ans.
```

TC: O(sqrt(N))

N = 100

Q. Magical no.

A magical no. is a no. that is a power of 5 or it can be formed by summing up unique/distinct powers of 5.

+ve power

+ve power

1	?	X	5^0. X not allowed.
1/5	?	X	5^{-1} X not allowed.
5	?	✓	5^1
25	?	✓	5^2
125	?	✓	5^3
10	?	X	$5^1 + 5^1$
			No
30	?	✓	$5^2 + 5^1 = 30$
150	?	✓	$5^3 + 5^2 = 150$
155	?	✓	$5^3 + 5^2 + 5^1 = 155$
31	?	X	$5^2 + 5^1 + 5^0 = 31$

Q. Find me the Nth magical no.

1st magical no. is 5.

15 = $5^1 + 5^1 + 5^1$ ✗

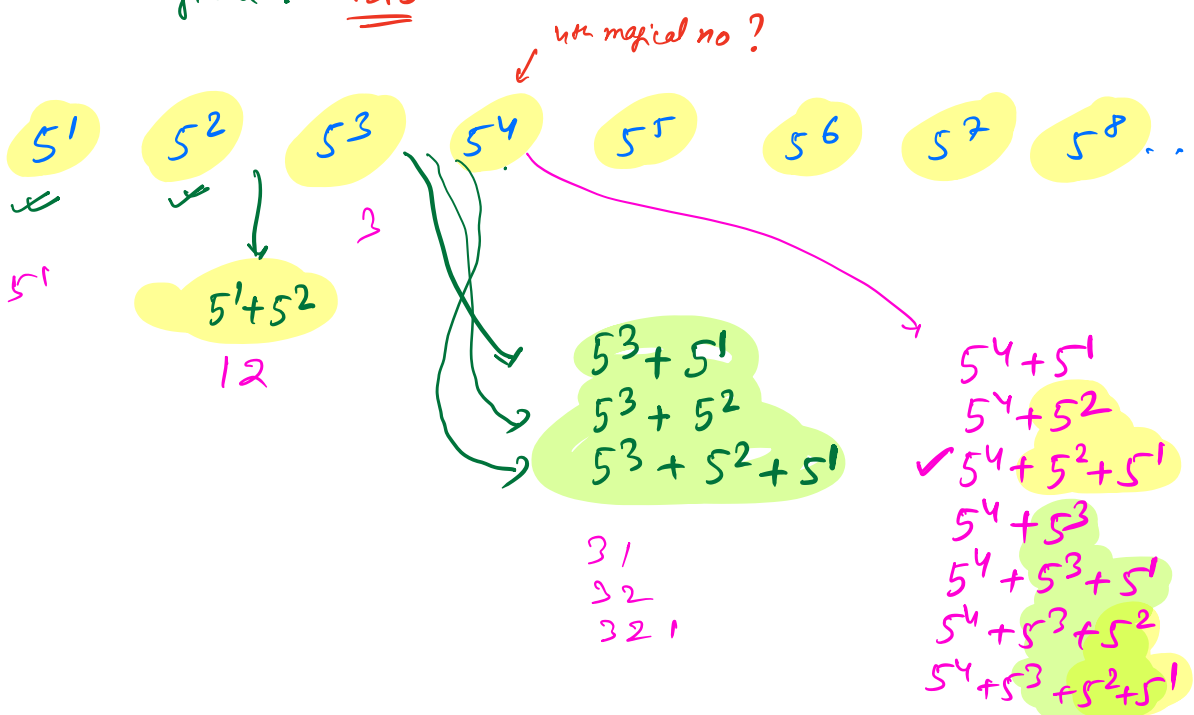
2nd magical no. is 25 5^2.

3rd magical no. is 30. $5^2 + 5^1$

4th magical no. is 125.

5th magical no. is 125+5 = 130.!

5th magical : 125



1st magical no. = 5^1

2nd " = $5^2 + 0$

3rd " = $5^2 + 5^1$

4th " = $5^3 + 0 + 0$

5th " = $5^3 + 0 + 5^1$

6th " = $5^3 + 5^2 + 0$

7th " = $5^3 + 5^2 + 5^1$

8th " = $5^4 + 0 + 0 + 0$

.
.
.
.
.

15th " = $5^4 + 5^3 + 5^2 + 5^1$

16th = $5^5 + 0 + 0 + 0 + 0 \ldots$

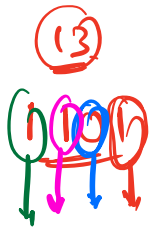
		4	3	2	1	
1st	5	0	0	0	1	$\leftarrow 5^1$ ✓
2nd	25	0	0	1	0	$\leftarrow 5^2$ ✓
3rd	30	0	0	1	1	$\leftarrow 5^2+5^1$ ✓
4th	125	0	1	0	0	$\leftarrow 5^3$
5th	130	0	1	0	1	$\leftarrow 5^3+5^1$
6th		0	1	1	0	$\leftarrow 5^3+5^2$
7th		0	1	1	1	$\leftarrow 5^3+5^2+5^1$
8th		1	0	0	0	$\leftarrow 5^4$
		1	0	0	1	$\leftarrow 5^4+5^1$
		1	0	1	0	$\leftarrow 5^4+5^2$
		1	0	1	1	$\leftarrow 5^4+5^2+5^1$
		1	1	0	0	$\leftarrow 5^4+5^3$
		⋮	⋮	⋮	⋮	⋮
15th		1	1	1	1	$\leftarrow 5^4+5^3+5^2+5^1$

Binary representation

N → 1 1 0 1 1 1 0 0 1

10 9 8 7 6 5 4 3 2 1

Nth magical no. = $5^{10} + 5^9 + 5^6 + 5^5 + 5^4 + 5^1$




```
def magicaN.( N ) :
    ans =0 , i= 1 :
    while (N > 0):
        ans = ans + (N %2) * pow(5,i)
        i++
        N= N//2
```

$5^3 + 5^1 + 5^4$

O(Log N)

```
    ans=0 , i=0
    lis=[ ]
    while (N>0):
        lis.append(N %2)
        N=N/2
    for ele in lis:
        ans = ans + ele * pow(5,i)
        i++
```

Break till 10:17 !

* Majority element.

Given an array of N elements, you have to find the majority element. If, a majority is not present, return -1

is majority.

{2, 2, 3, 1, 1, 1} ← 1 is occuring 50% only

1 occurs max no. of times.

No. should be occurring >50% of the times.

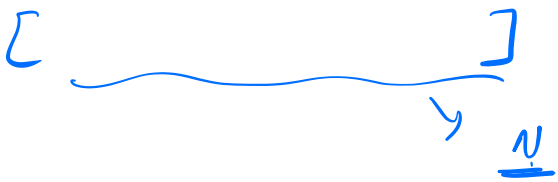
100 people.

33 people. , 40 people, 27 people.

51 people.

majority.

* How many majority elements can be present in an array of size N?



N/2 + 1

Assume, there are 2 majority elements.

atleast $\frac{N}{2}+1 + \frac{N}{2}+1$

N+2 element cannot be present in an array of size N.

Q. arr = [3 1 3 3 3 2 4 1 3]

Find the majority element.

TC: O(N)

Brute force:

Count frequency of each no.

SC: O(N) ← Storing the count of each no.

SC: O(1) ←

Find the majority in constant space. & O(N) Time.

* Remove 1 BJP and 1 Congress/minority party.
  → Yes

✗ Remove 2 BJP.
  → No ✗

✓ Remove 1 Congress / 1 AAP (2 minority) = Yes

Total people

* N.
* $N-2$

Majority

$\frac{N}{2} + 1$

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

$= \frac{N}{2}$

N

$\frac{N}{2}+1$ in Majority, $\frac{N}{2}-1$ in Minority.

After removing 2 persons:

$\frac{N}{2}$ in majority, $\frac{N}{2}-2$ in minority

>50%

Or $\frac{N}{2}+1$ in majority, $\frac{N}{2}-3$ in minority.

>50%

$N-2$ people

$N-2$

↳ Both in majority XX

Both in minority

1 in majority, 1 in minority

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

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

$= \frac{N}{2}$

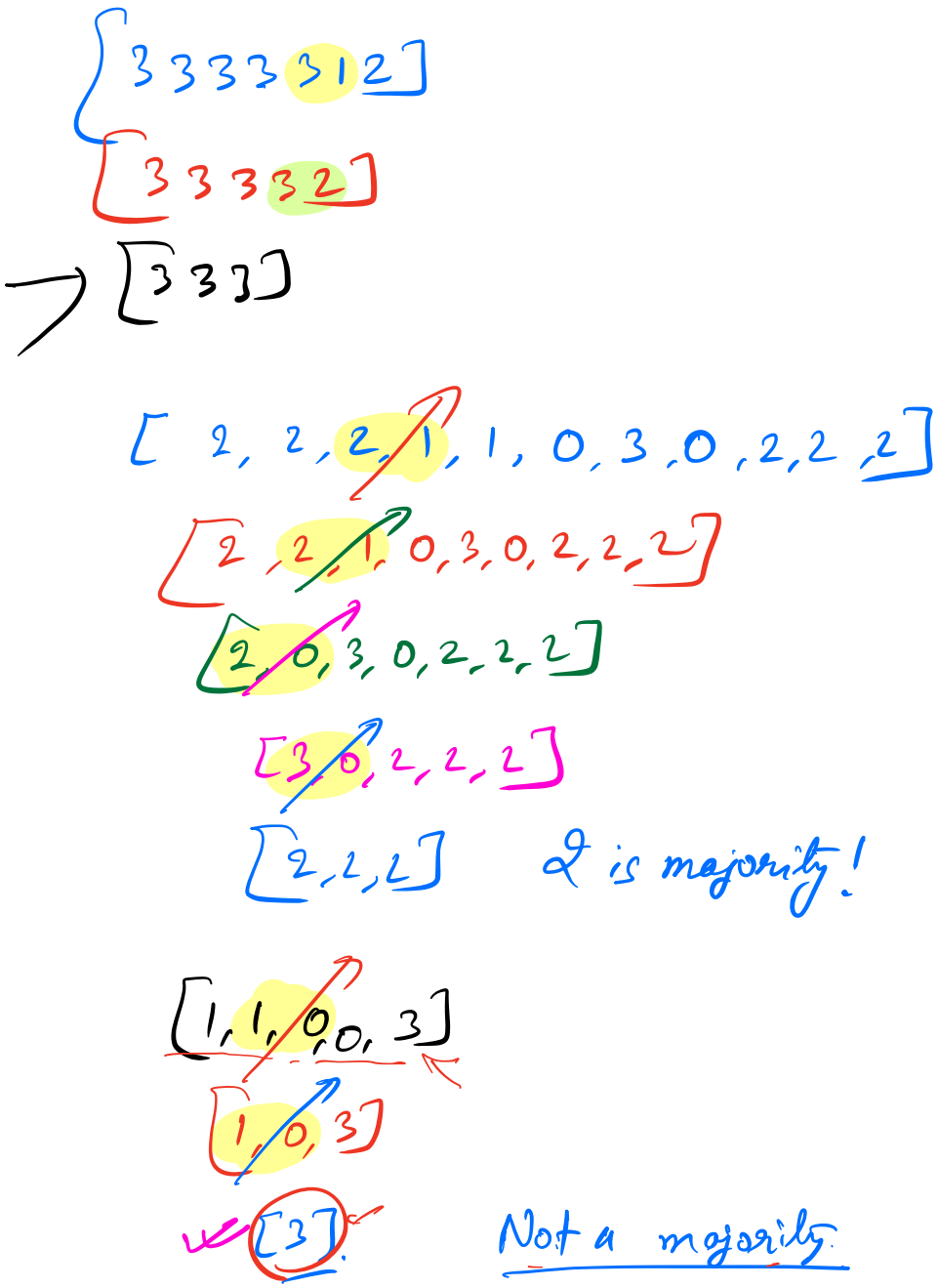
arr = [3 1 3 3 3 2 4 1 3]

[3 3 3 2 4 1 3]

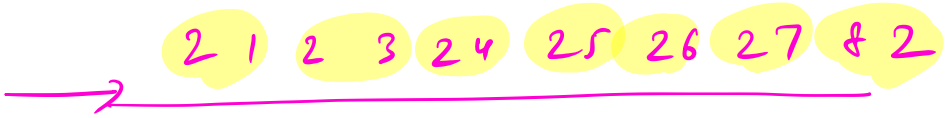
[3 3 4 1 3]

[3 1 3]

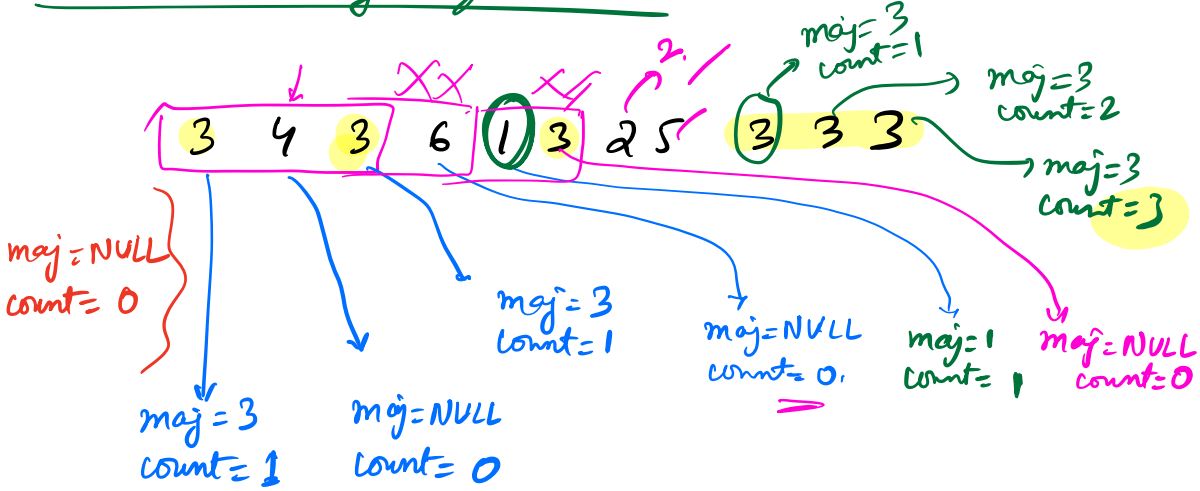
[3] ← Majority!

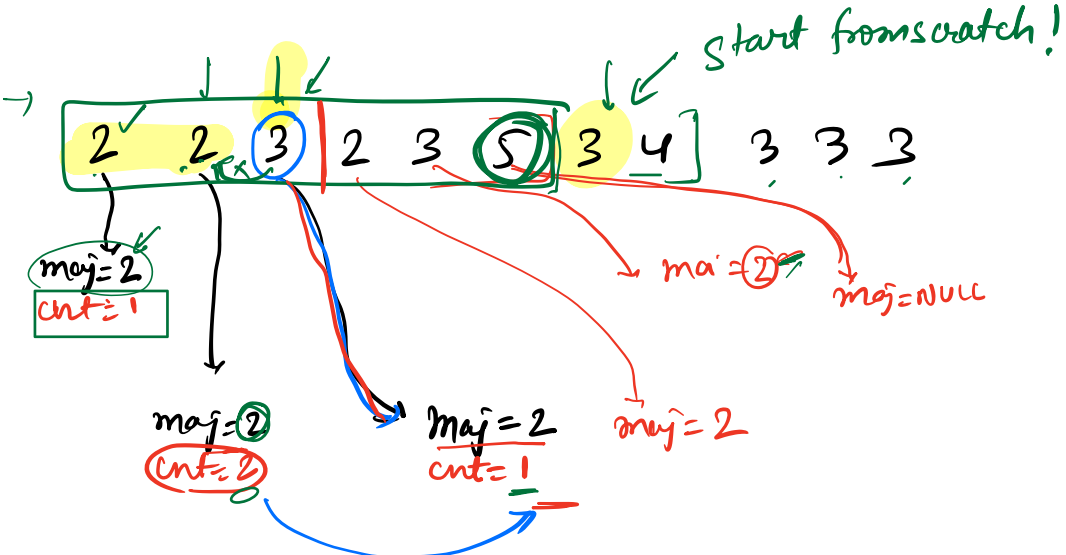


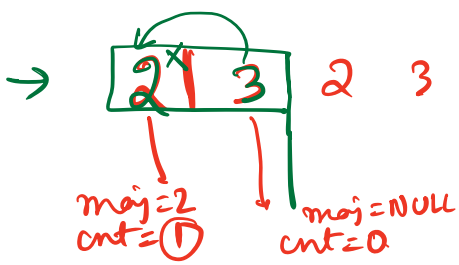
* Check whether that element is actually in majority.



Moore's voting algorithm.







* Iterate the array from left to right.

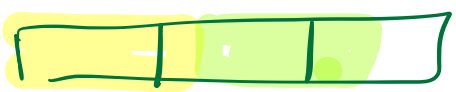
* Maintain 2 variables. maj and cnt.

* At the end, 'maj' is your candidate.

* Check whether the candidate is actually in majority.

Follow up:

How to find N/3 majority element?



100

40 40 20

Hint.

maj1 count1 maj2 count2

HW.

A: "532198320102"

(3)

$(2 \times 10^0) \% B$

$+ (0 \times 10^1) \% B$

$+ (1 \times 10^2 \% B)$