Algorithm Problems:-

(1) **Majority Element I** → (M1) $O(n^2)$ brute
(M2) $O(n \log n)$ Hashing

find the element that appears more than $\frac{N}{2}$ times in the array. ie ($> \frac{N}{2}$ times)

(M3) ↳ **Moore's Voting Algorithm**:-

eg:-

arr[ ] = [7 7 5 7 5 | 5 7 5 5 7 7 5 5 5 5]

(S1) consider 7 to be the majority elem

(S2) cnt = 0

(cnt ++ if majority elem appears & -- if it doesn't)

if 7 is majority elem then cnt > 0, bcz if it appears $> \frac{N}{2}$ times. other elem combined can't cancel it

elem = 7

cnt = ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~2~~ ~~1~~ 0

Observation 1)

In this part of array 7 ≠ majority elem. nor is any other elem appearing in this part

bcz cnt = 0 & ~~means~~ 7 appears exactly half len of this part & so other cannot appear >half this len.

bcz

Observation 2)

when $\boxed{cnt = 0}$

↓

then current subarray doesn't contain any element which is majority elem is current subarray.

↳ (So discard it from consideration)

Conclusion

↳ if we divide the array into multiple parts (such that cnt of each part = 0)

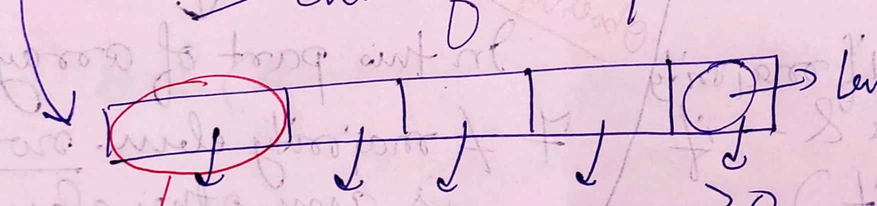
↳ (is of even len) or (1 len)

&&

if there is a majority elem.

(there has to be a part with cnt > 0)

else cnt of all parts == 0.



0 ⇒ best case (if elem = 2) in this part

both elem appear half len of subarr times

& net = 0

arr[] = [~~7~~ ~~7~~ ~~5~~ ~~7~~ ~~5~~ ~~1~~ 5 7 5 5 7 7 5 5 5 8]

elem = ~~7~~ ~~5~~ ~~5~~ 8

cnt = ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

S3

[nxt considers this to be majority elem.

[if cnt = 0 in subarr ⇒ no elem in subarr is majority elem. bcz 'elem' appears half times len of subarr]

S4 (cnt > 0 && array exhausted so this is the ~~elem~~ last stored is majority elem.)

S5 Reiterate array to verify if this is majority elem.

→ But if cnt = 1 at the last part (with len = 1)

↓

There could be that no other elem could nullify the elem ~~to make it~~

↳ So can not just call it majority elem.

so