Bayer - moore algorithm

counting majority element in an array

$n = \text{len}(\text{array})$

$m \in$ array is majority if m.freq = # of times m occurs in array $> \lfloor \frac{n}{2} \rfloor$

(ex)

[1, 2, 1, 1] ← 1 majority

[1, 2, 1, 2] ← no majority

[1, 2, 1, 2, 1] ← 1 majority

[1, 2, 1, 2, 3] ← no majority

Using a dictionary to store each value's freq → can do it in linear time. $O(n)$

and linear space $O(n)$

But we can do it in $O(n)$ time and

$O(1)$ space! ← Bayer - moore
"streaming algorithm"

Visualizing

Battle field

count = # of majority army soldiers in battle field

candidate = potential majority army

count = 0
candidate = None
no armies, no count

scan each num in array:

→ if count = 0,
candidate = num

if num enters an empty battlefield, his army becomes candidate for majority

→ if num is candidate → count += 1
if num not candidate → count -= 1

If majority candidate army soldier joins, he adds to its strength
If soldier of other armies join, he kills a majority army soldier

The candidate when all num in array are scanned → majority element / army

[2 things : really majority if array admits a majority element
if array has no majority element, double check!

Formal proof of correctness

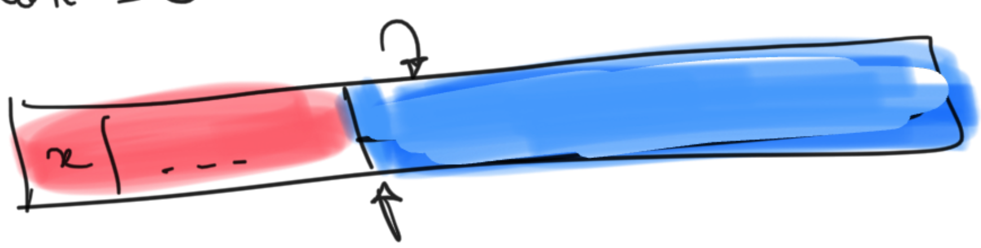
→ Argue by induction

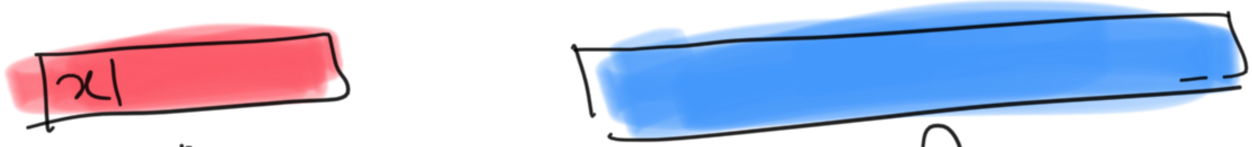
→ if count $\neq 0$ ever ... } base cases

if [*] array size = 1

→ Then argue by looking at first place

count = 0





$\Downarrow$

x.freq in

= len( )/2

and len( ) = even

algo returns m

$\Rightarrow$ m = majority element of

by induction.

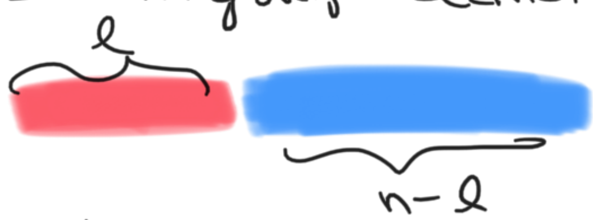
Case 1

if has a clear majority

Case 2 : no clear majority in blue ??

Let us only look at case 1

show m = majority element in



2 sub cases

$m = x$

$m \, != x$

m.freq

let $y \neq m$

$= x.\text{red freq} + m.\text{blue freq}$

$= \ell/2 + m.\text{bluefreq}$

$>^* \ell/2 + \left\lfloor \frac{n-\ell}{2} \right\rfloor$

$\geq \lfloor n/2 \rfloor$

we are in case 1

Argue

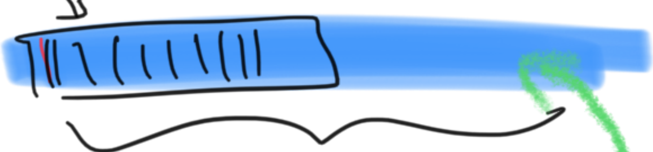
$y.\text{freq} = y.\text{red freq} + y.\text{blue freq}$

$\leq \lfloor n/2 \rfloor$.

$$\begin{cases} y.\text{redfreq} \leq \ell/2 \\ y.\text{bluefreq} \leq \left\lfloor \frac{n-\ell}{2} \right\rfloor \end{cases}$$

because

m has $> \left\lceil \frac{n-\ell}{2} \right\rceil$



$\frac{n-\ell}{2}$

y has to be $\leq \left\lfloor \frac{n-\ell}{2} \right\rfloor$

$y.\text{freq} \leq \left\lfloor \frac{n-\ell}{2} \right\rfloor + \ell/2$

...

n even

as ℓ even

$\left\lfloor \frac{n-\ell}{2} \right\rfloor = \frac{n-\ell}{2}$

so

$y.\text{freq} \leq \frac{n}{2}$

if $y.\text{freq} < \frac{n}{2}$, done,

else if $y.\text{freq} = \frac{n}{2}$,

no clear majority in array.

n odd $= 2k+1$

as ℓ even

$\left\lfloor \frac{n-\ell}{2} \right\rfloor = \left\lfloor \frac{2k+1-\ell}{2} \right\rfloor$

$= k - \left(\frac{\ell}{2}\right)$

so

$y.\text{freq} \leq k - \left(\frac{\ell}{2}\right) + \left(\ell/2\right)$

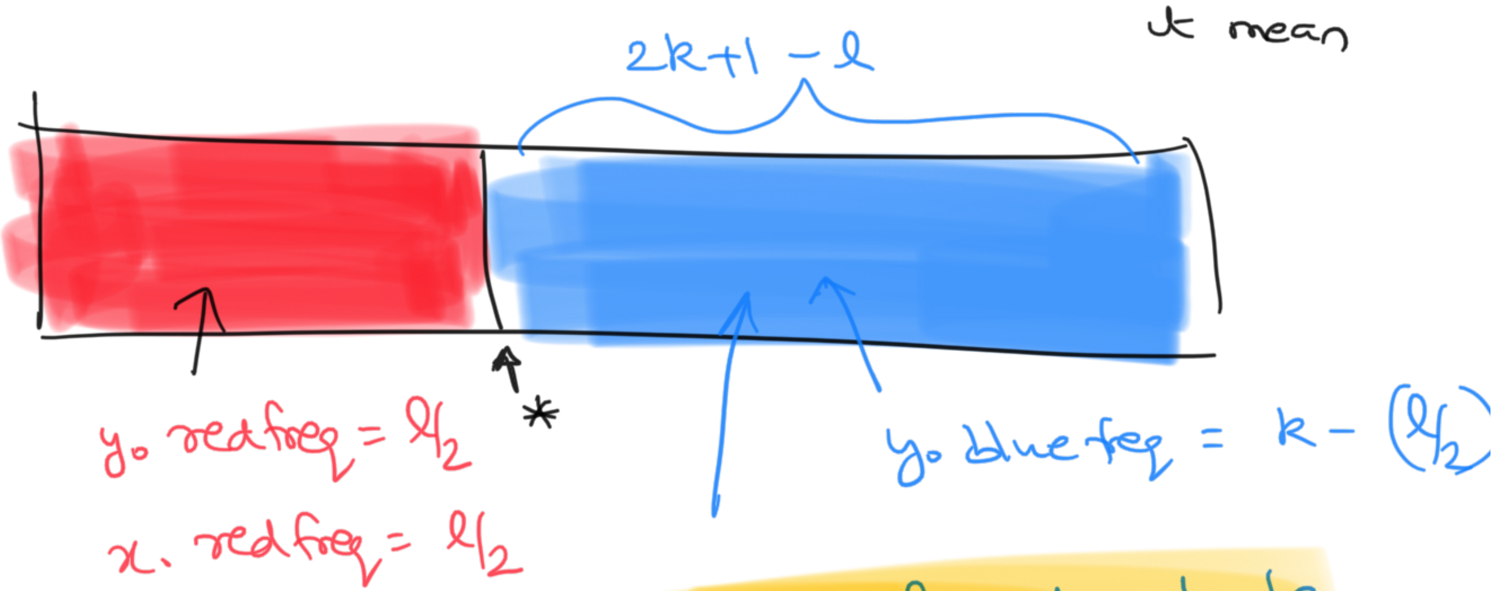
$= k$

So algo outputs some thing

$\lfloor n/2 \rfloor = k \ldots$

If $y.freq < k$, done

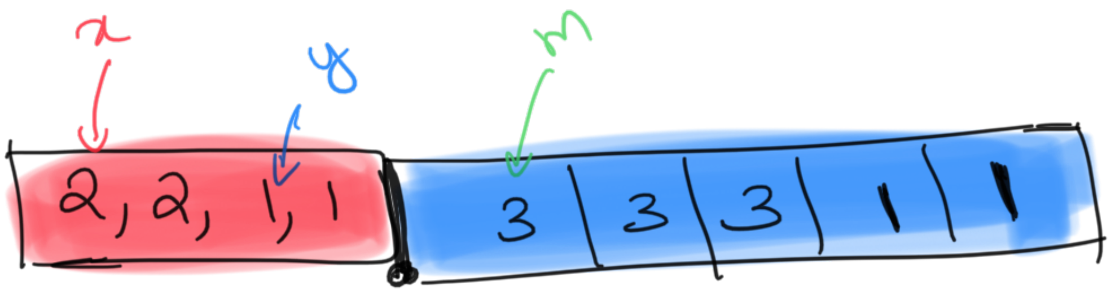
If $y.freq = k, \ldots$ what does it mean



m. bluefreq has to be $k - \left(\frac{\ell}{2}\right) + 1$ as m is clear majority ...

as case 1

eg



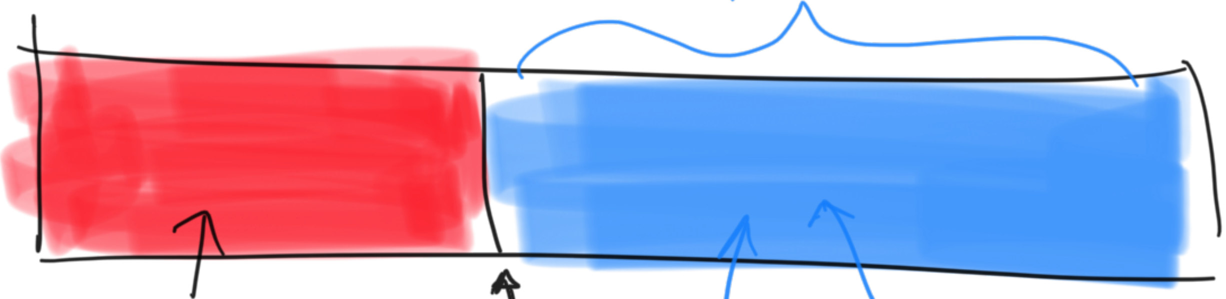
Count	Candidate	num
0	None	
1	2	2 ←
2	2	2 ←
1	2	1 ←
0	2	1 ←
0	2	3 ←
1	3	

2	3	3	←
3	3	3	←
2	3	1	←
1	3	1	←

outputs candidate = 3 , but note there is no clear majority here!! in full array, so safe...

Let us go back to picture

$2k+1-\ell$



*

y. redfreq $= \ell/2$

x. redfreq $= \ell/2$

y. bluefreq $= k - (\ell/2)$

m. bluefreq has to be $k - \left(\frac{\ell}{2}\right) + 1$ as m is clear majority...

as case 1

we'll show  has no clear majority!

$y \neq x$

$y : \quad \ell/2 + k - \ell/2 = k$

$y = x$

$y : \quad \ell/2 + k - \ell/2 = k$

$m : \quad k - (l/2) + 1$

$x : \quad l/2$

$n : \quad 2k+1 = \#y + \#x + \#m$

so no clear majority!

as nr $> k$ for clear majority

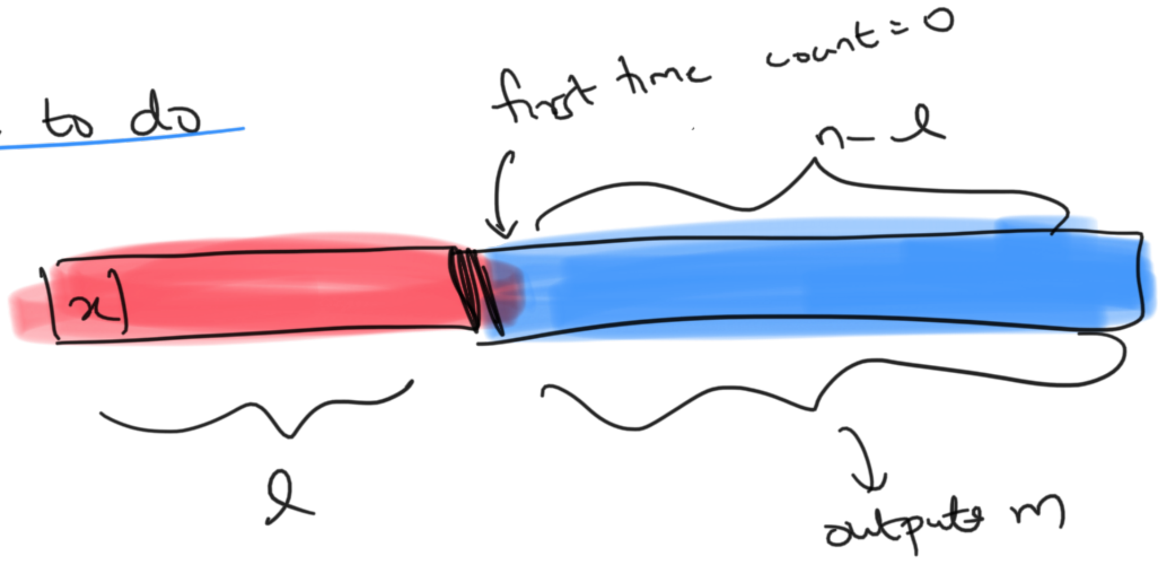
$m : \quad k - (l/2) + 1$

+ at most $l/2$ in red

claim: anyway y can't be clear majority

as y.freq $= k$ not $> k$ ....

Left to do



what if  has no clear majority but  has a clear majority ..... why is 'm' the correct answer then?