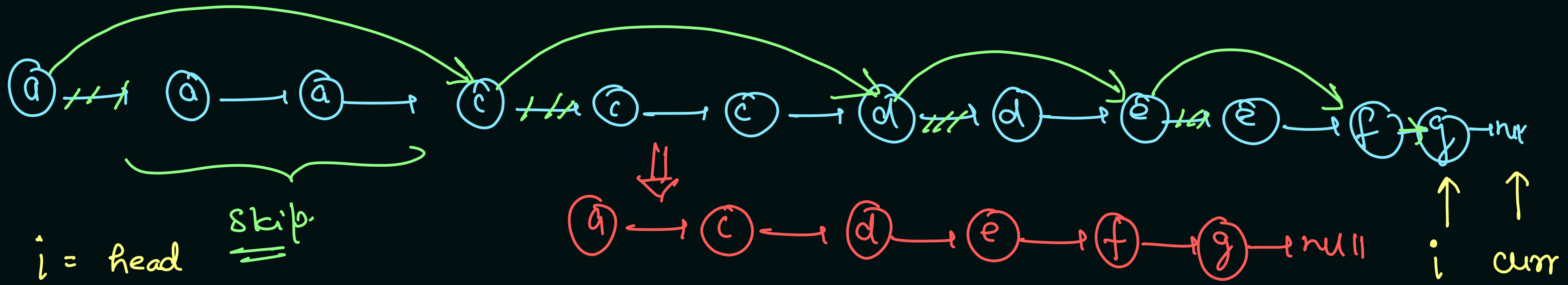
Remove duplicates:→ linked list is sorted → duplicates encounter together.



i = head

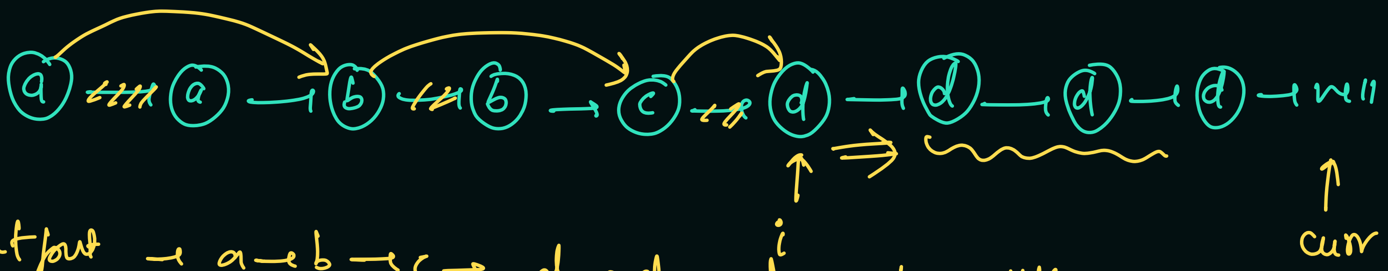
curr = head.next

i.val == curr.val	i.val != curr.val

curr = curr.next	i.next = curr
	i = curr
	curr = curr.next

end

i.next = null



output → a→b→c→ d→d →d →d→null

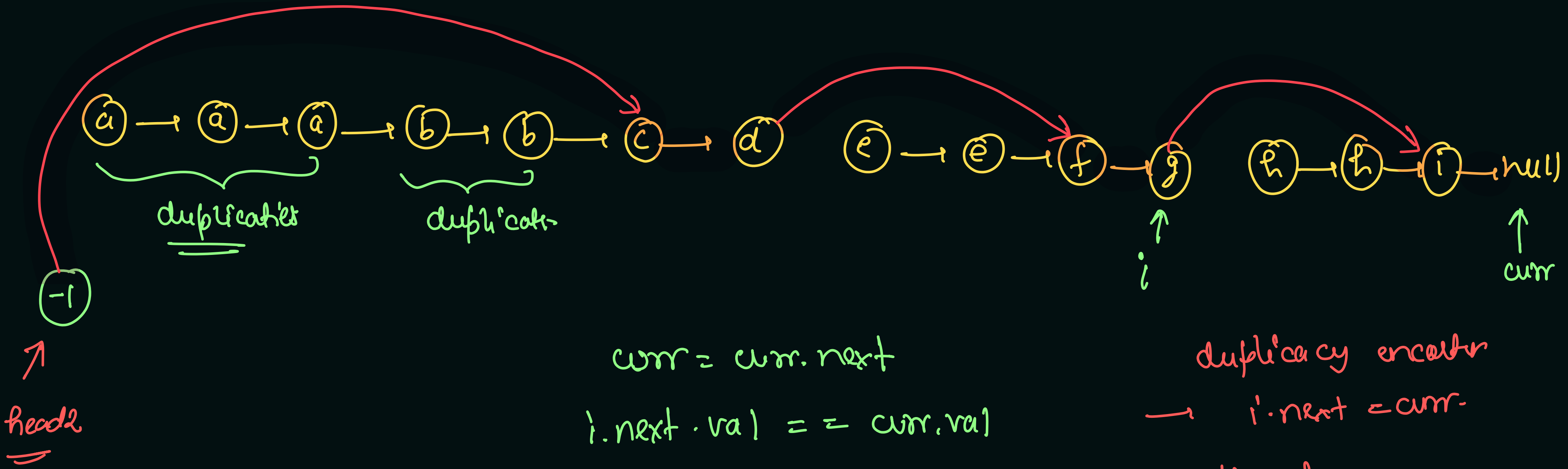
Remove all duplicates from sorted linked list :→

output →

$$\begin{tikzcd} c \arrow[r] & d \arrow[r] & f \arrow[r] & g \arrow[r] & i \arrow[r] & null \end{tikzcd}$$

input

$$\begin{tikzcd} a \arrow[r] & a \arrow[r] & a \arrow[r] & b \arrow[r] & b \arrow[r] & c \arrow[r] & d \arrow[r] & e \arrow[r] & e \arrow[r] & f \arrow[r] & g \arrow[r] & h \arrow[r] & h \arrow[r] & i \arrow[r] & null \end{tikzcd}$$



curr = curr.next

i.next.val == curr.val

duplicacy encounter

→ i.next = curr.

otherwise

i = i.next

return head2.next

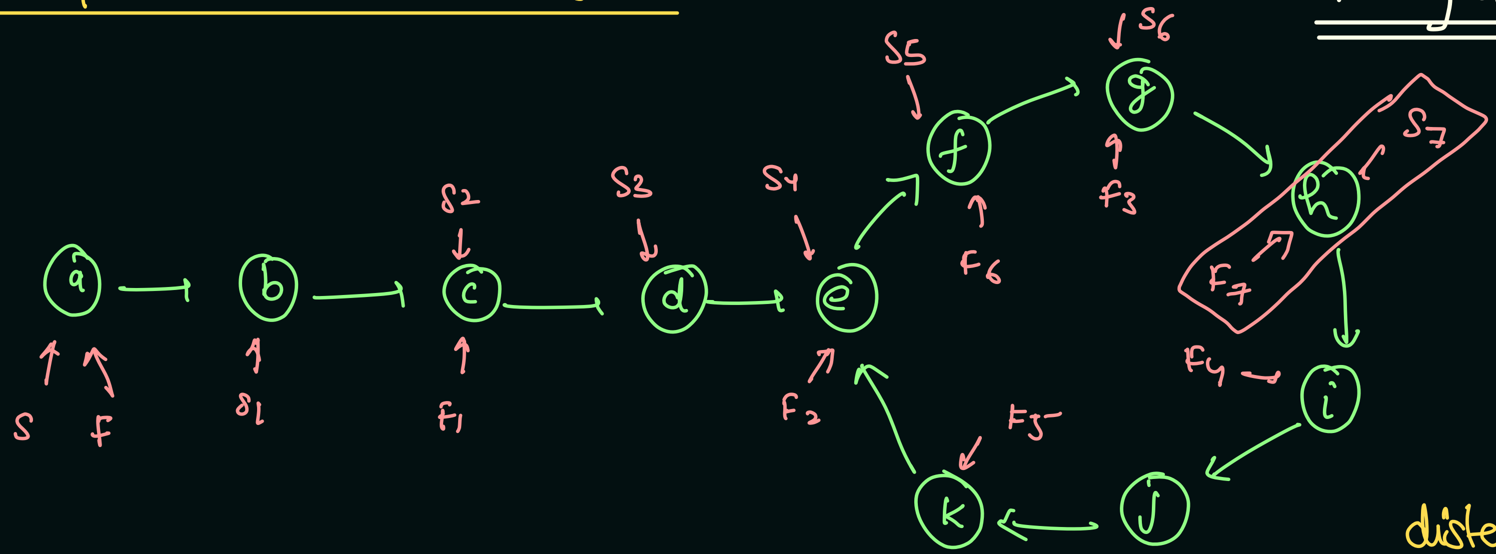
head2

output.

$$\begin{tikzcd} -1 \arrow[r] & c \arrow[r] & d \arrow[r] & f \arrow[r] & g \arrow[r] & i \arrow[r] & null \end{tikzcd}$$

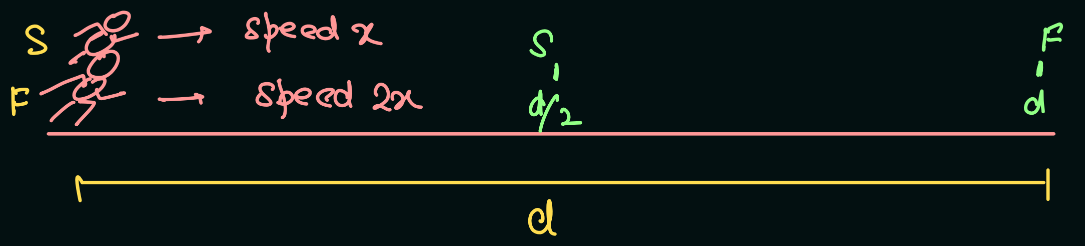
Floyd Cycle Detection Algorithm

Is Cycle present in Linked List :→



distance = speed × time

Straight line track



distance travel by 'S' in 't' time.

$d = x \times t = xt$

distance travel by 'f' in 't' time

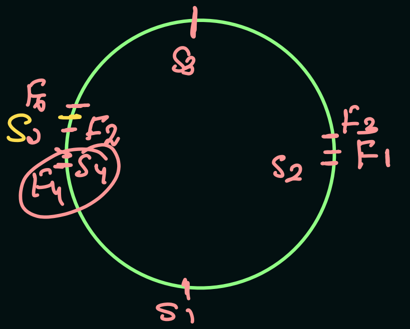
$d = 2x \times t = 2xt$

If path is cyclic →

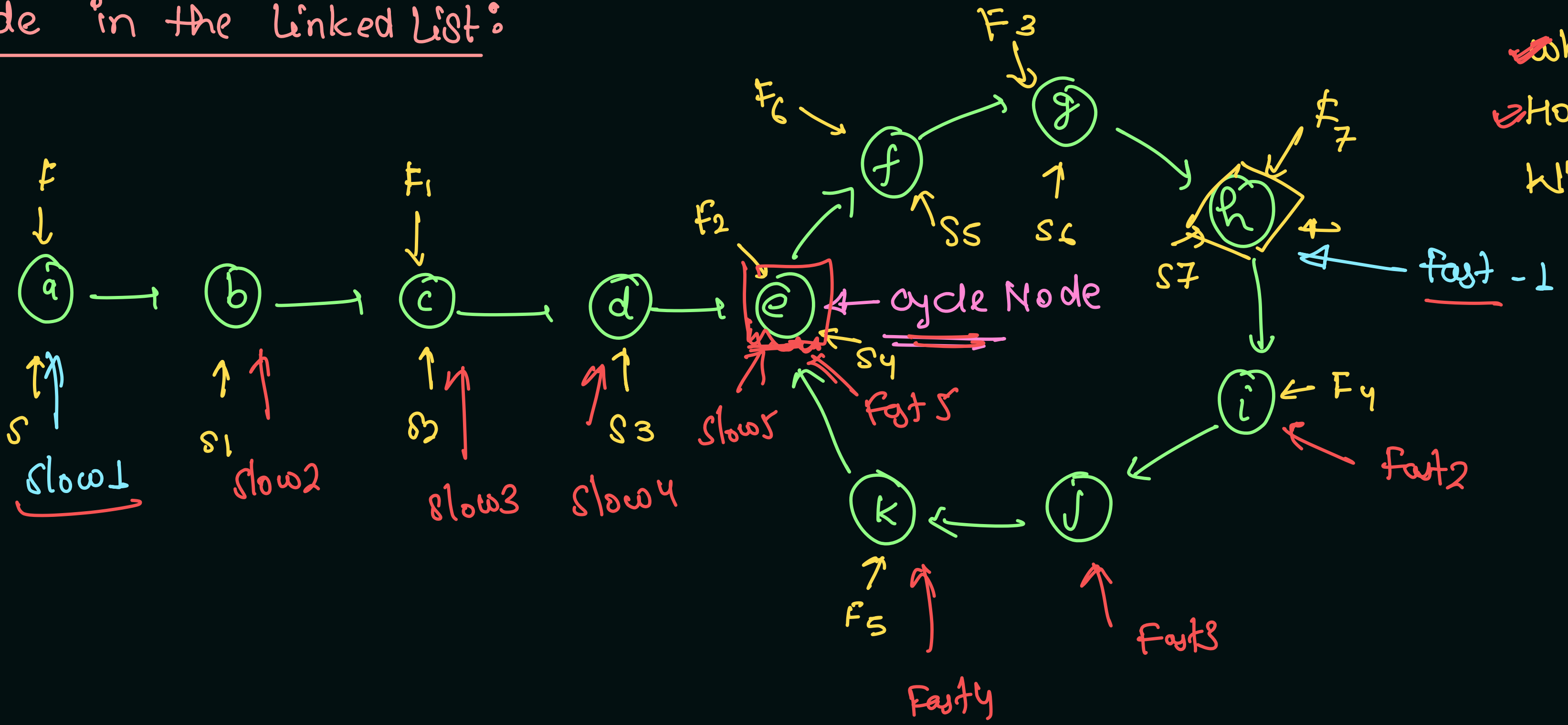
then certainly slow & fast encounter at a some pointer.

because fast have 2x speed & slow have 'x' speed

→ cyclic path, slow & fast meet at a common point



Cycle Node in the Linked List:

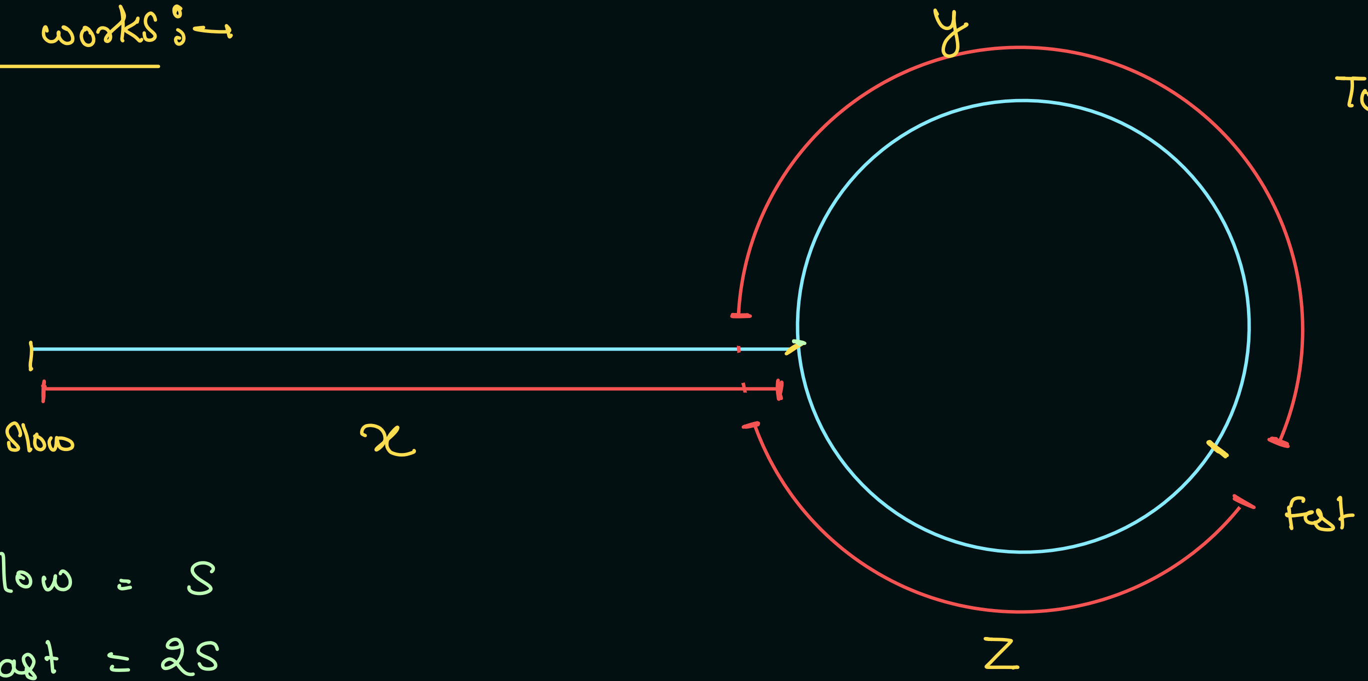


- What ? ✓
- How ? ✓
- Why ? ✓

What

1. Find it is cyclic or Not.
2. Place fast at common meeting point & slow at head.
3. Move slow & fast at same speed. (x)
4. the point where slow & fast meet is starting point of cycle.

Why Algorithm works :→



To prove :

$x = z$

Speed of slow = S

Speed of fast = 2S

Time take by slow & Fast = t

distance travel by slow = St = x+y ⇒ $t = \frac{x+y}{s}$ —— ①

" " " Fast = 2st = x + m(y+z) + y ⇒ $t = \frac{x + m(y+z) + y}{2s}$ —— ②

we know that ① & ② is equal.

$$\frac{x+y}{\cancel{s}} = \frac{x + m(y+z) + y}{2\cancel{s}}$$

$$2x + 2y = x + m(y+z) + y$$

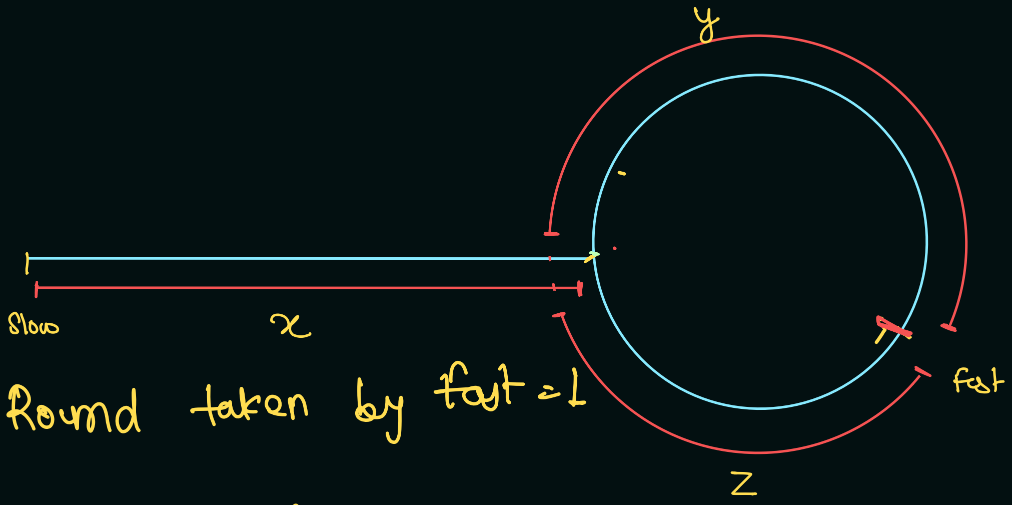
$x + y = m(y+z)$

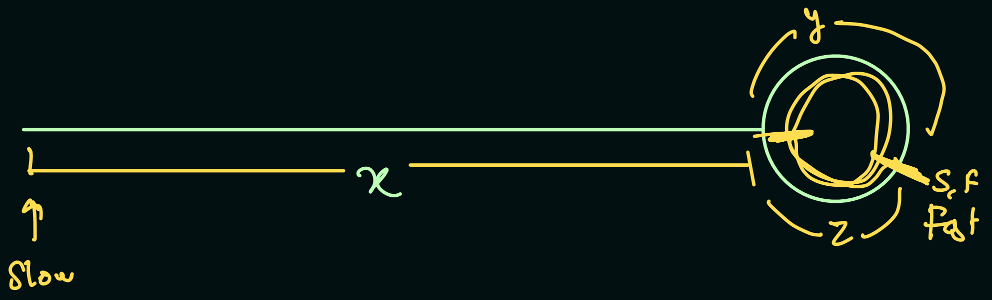
$x = m(y+z) - y$, $(\underline{y+z})$ add & subtract in R.H.S ,

$x = m(y+z) - (y+z) + (y+z) - y$

$x = (m-1)(y+z) + \cancel{y} + z - \cancel{y}$

$$\boxed{x = \underline{(m-1)}(\underline{y+z}) + z}$$ → How Round is helping in equality





Round taken by fast = 1

$x = (1-1)(y+z) + z$

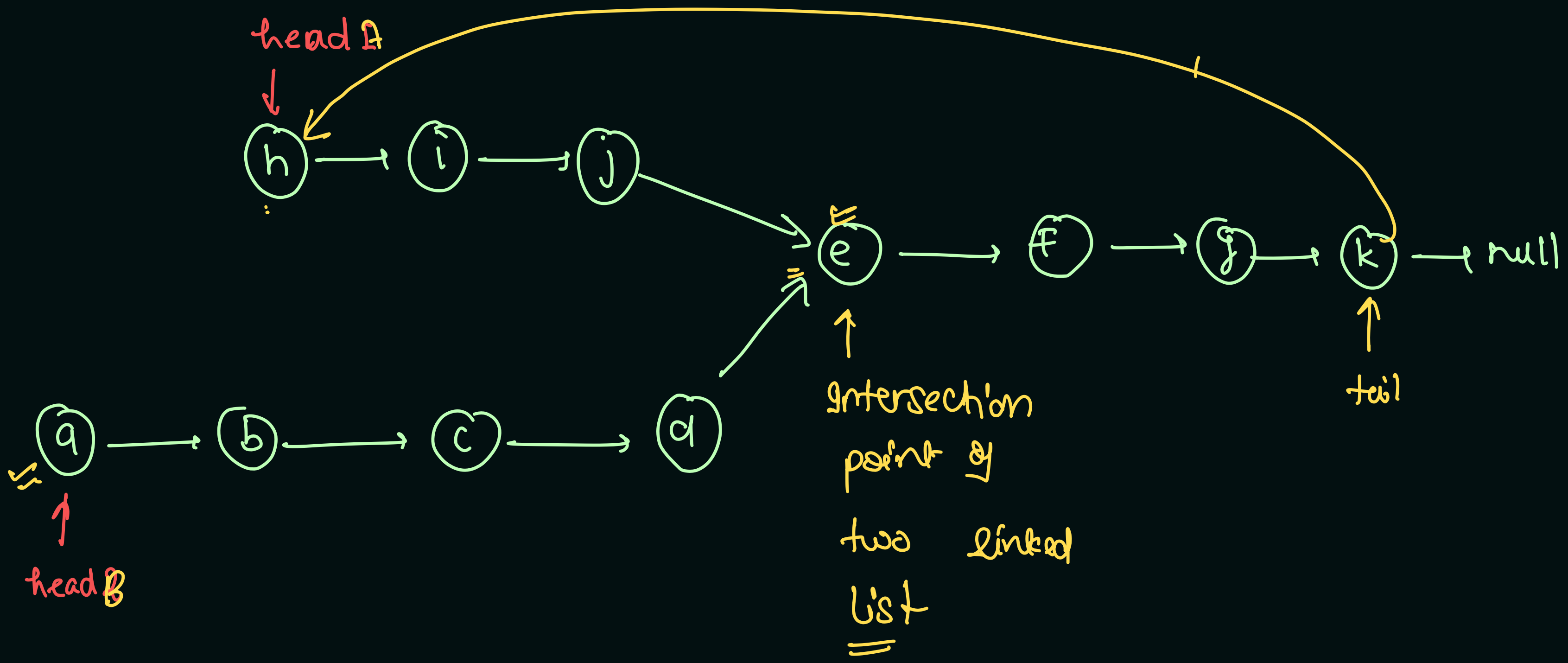
$x = 0 + z$

$\Rightarrow [x = z]$

$\underline{m=1}$ [Round = 1] ⇒ $\boxed{x = z}$

Intersection Node of two linked list :

Method 1 – using Floyd Cycle detection Algorithm



① Move from headA and find tail

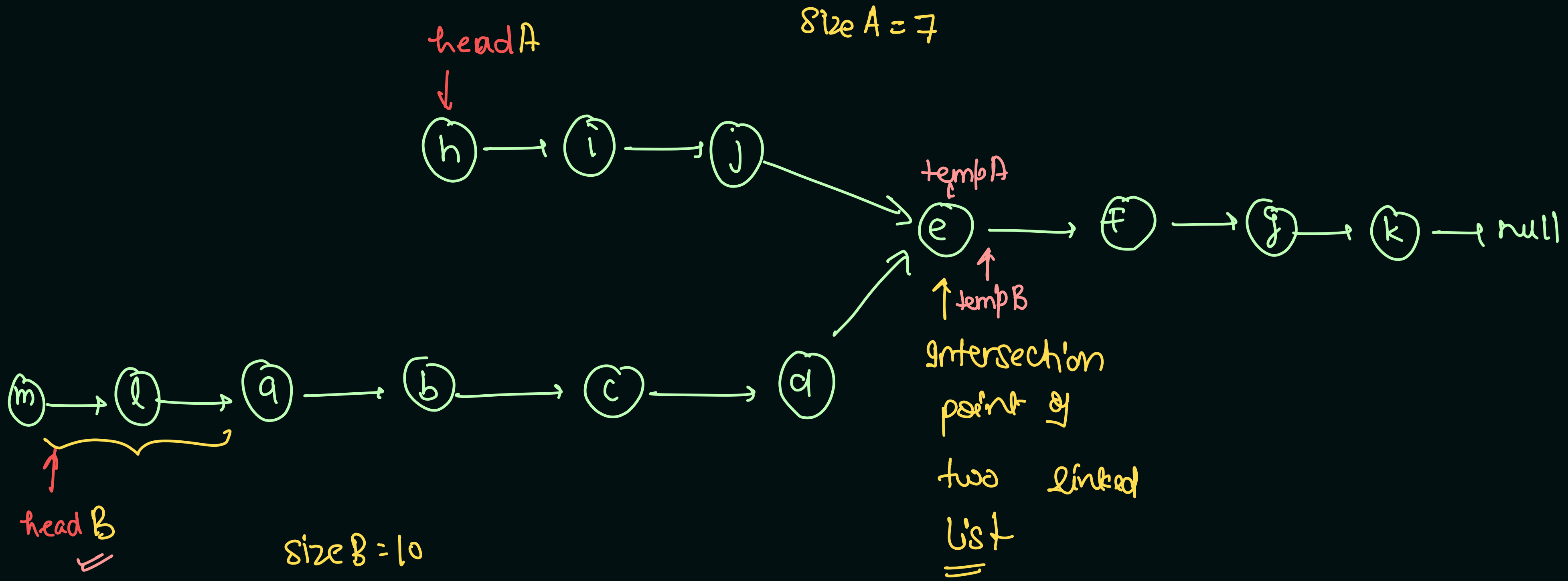
② connect tail. next to head A

③ Find starting node of cycle from head B.

④ If node present return that node otherwise null

⑤ Before returning result retain original structure.

Method 2 → using Difference Method:



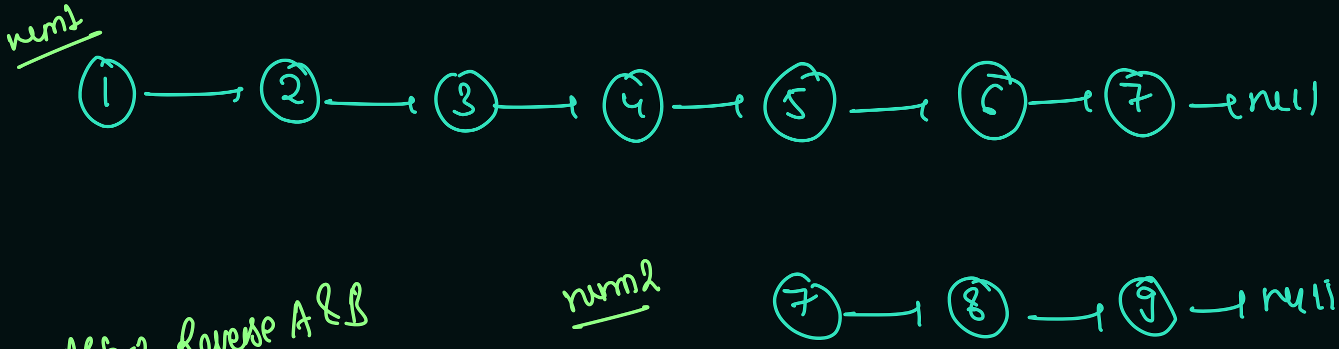
① iterate Extra Node Either in head A or head B.

② Move tempA & tempB simultaneously.

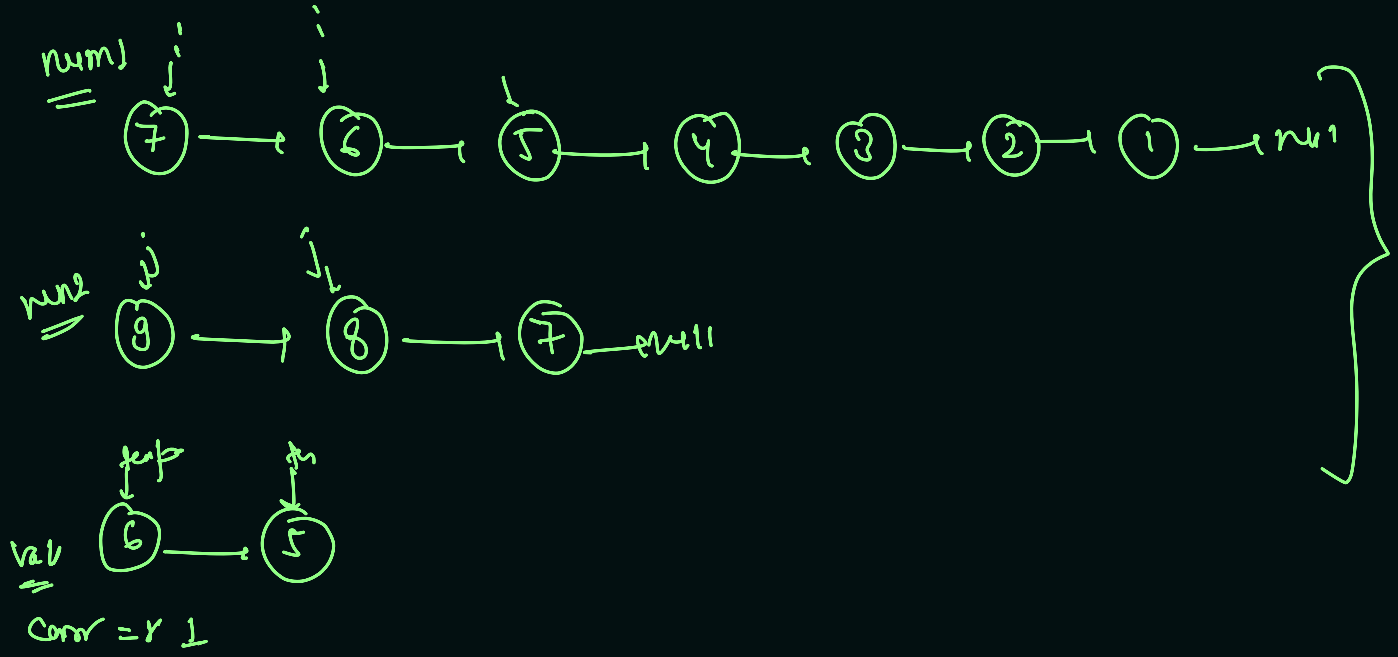
③ If encounter at same point, then that point is my result

otherwise no intersection point.

add two linked list :



After Reverse A & B



1. convert number into integer & solve.
2. store number in array & solve.
3. Recursion → space
   Equivalent to space of array

```
1 2 3 4 5 6 7
        7 8 9
-------------
1 2 3 5 3 5 6
```