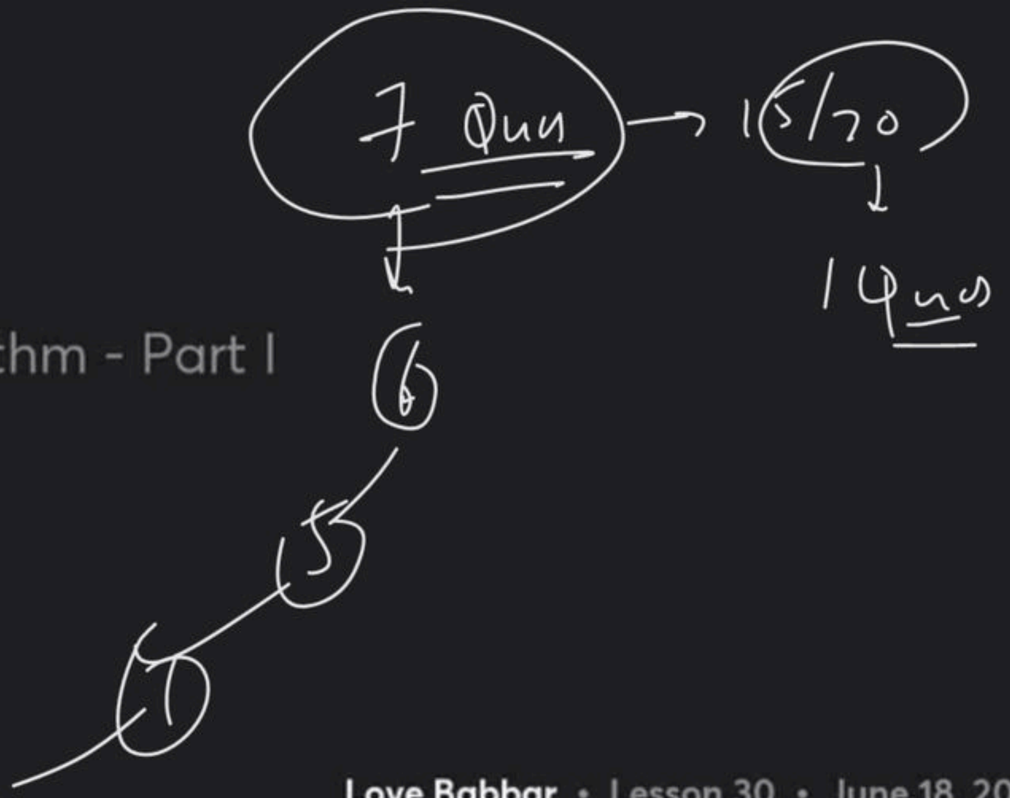
Linked List - II

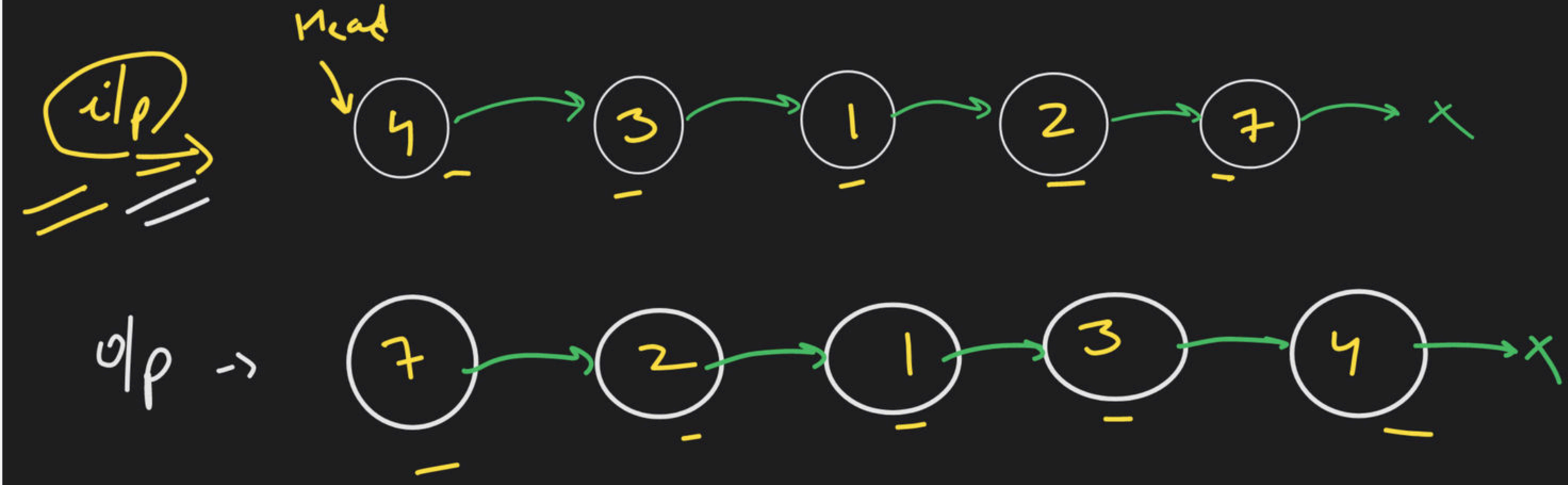
Foundation Course on Data Structures & Algorithm - Part I

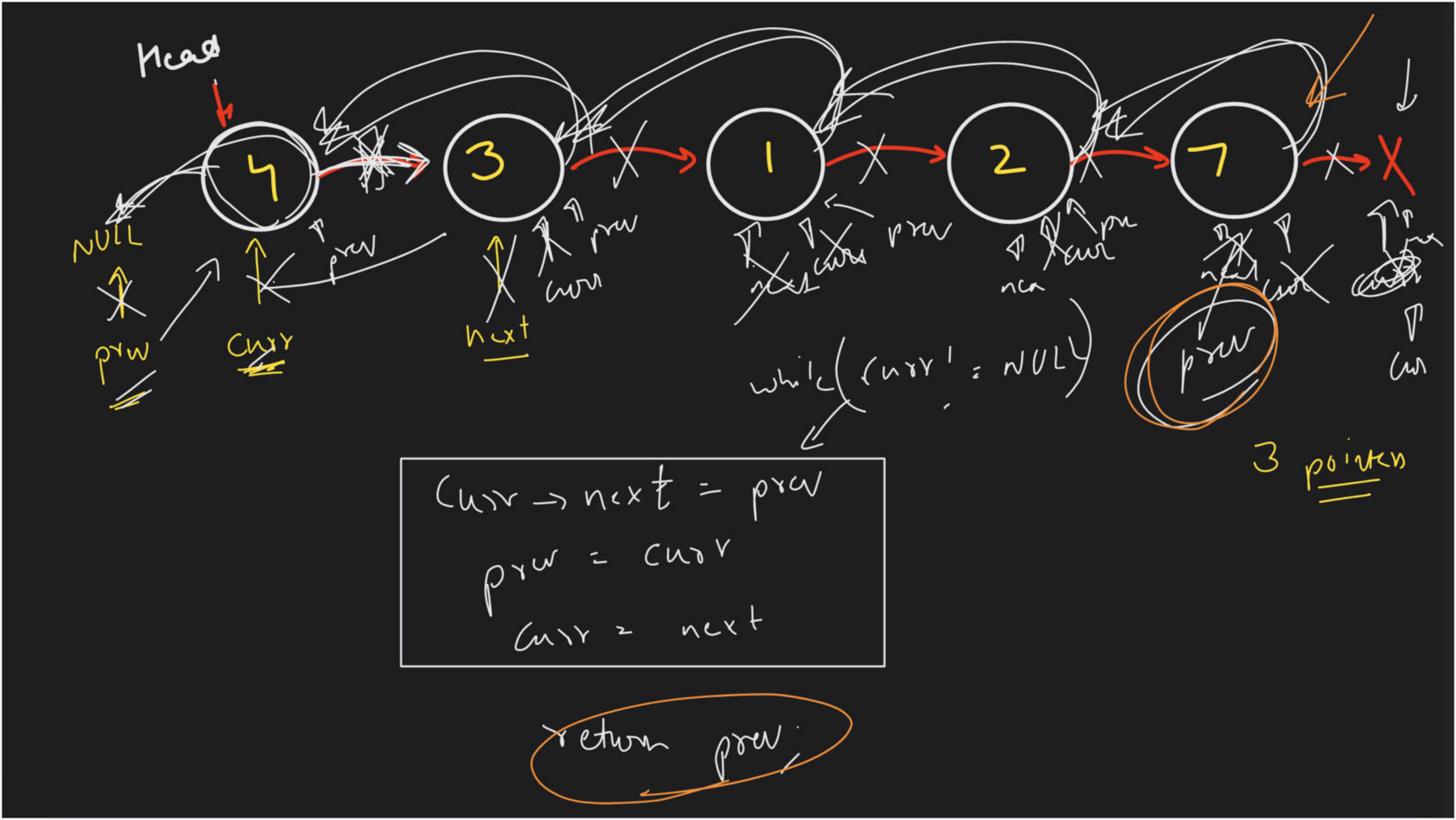


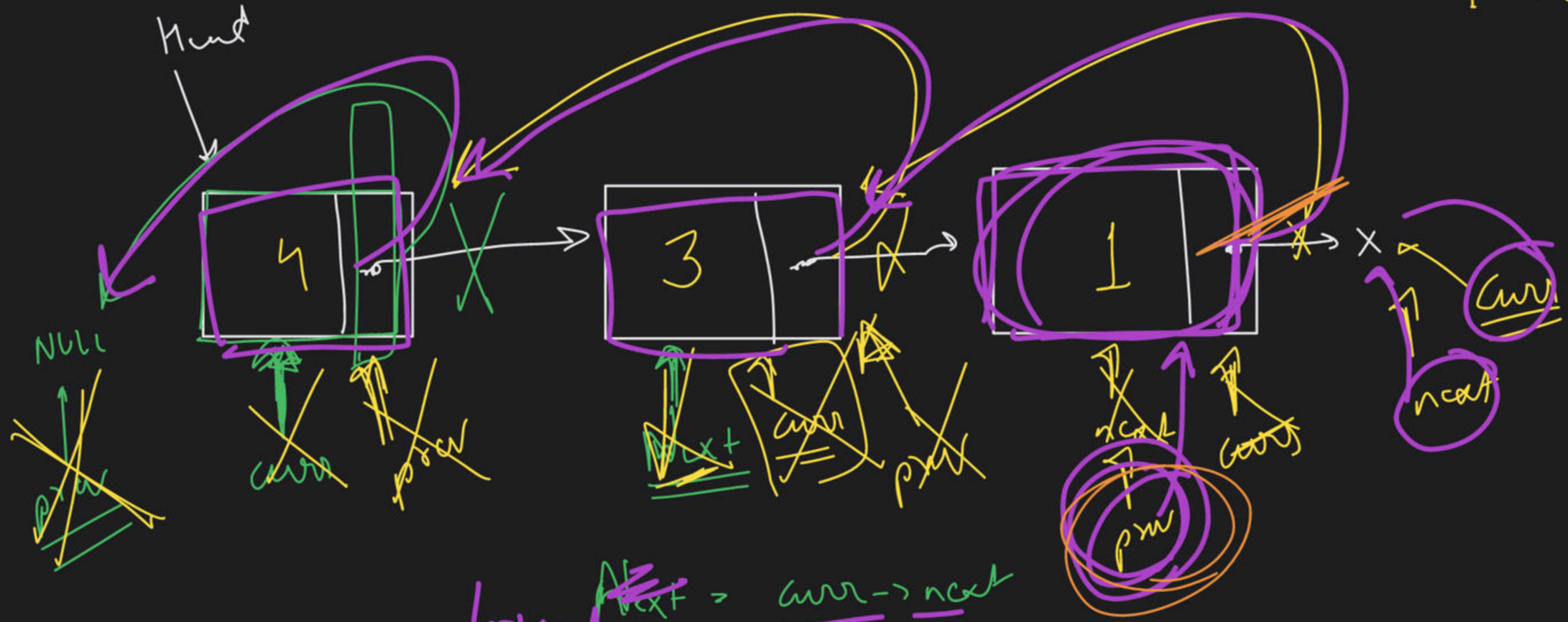
Love Babbar • Lesson 30 • June 18, 2022

Linked List — Reverse

easy




Head
4
3
1
2
7
X
NULL
prev
curr
next
while(curr != NULL)
curr->next = prev
prev = curr
curr = next
3 pointers
return prev;



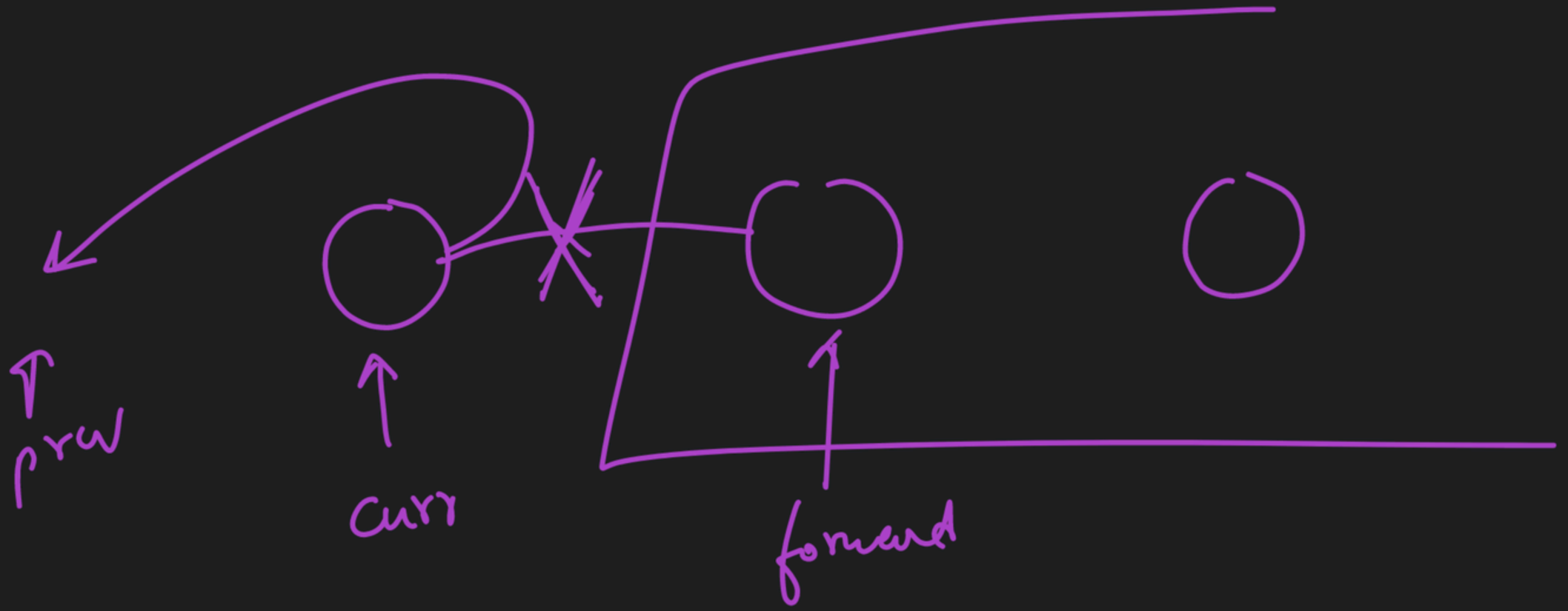
forward = curr->next

curr->next = prev

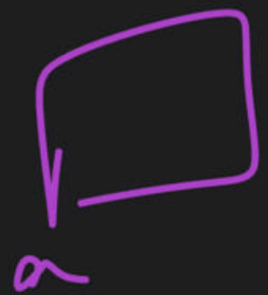
prev = curr

curr = forward

return prev;

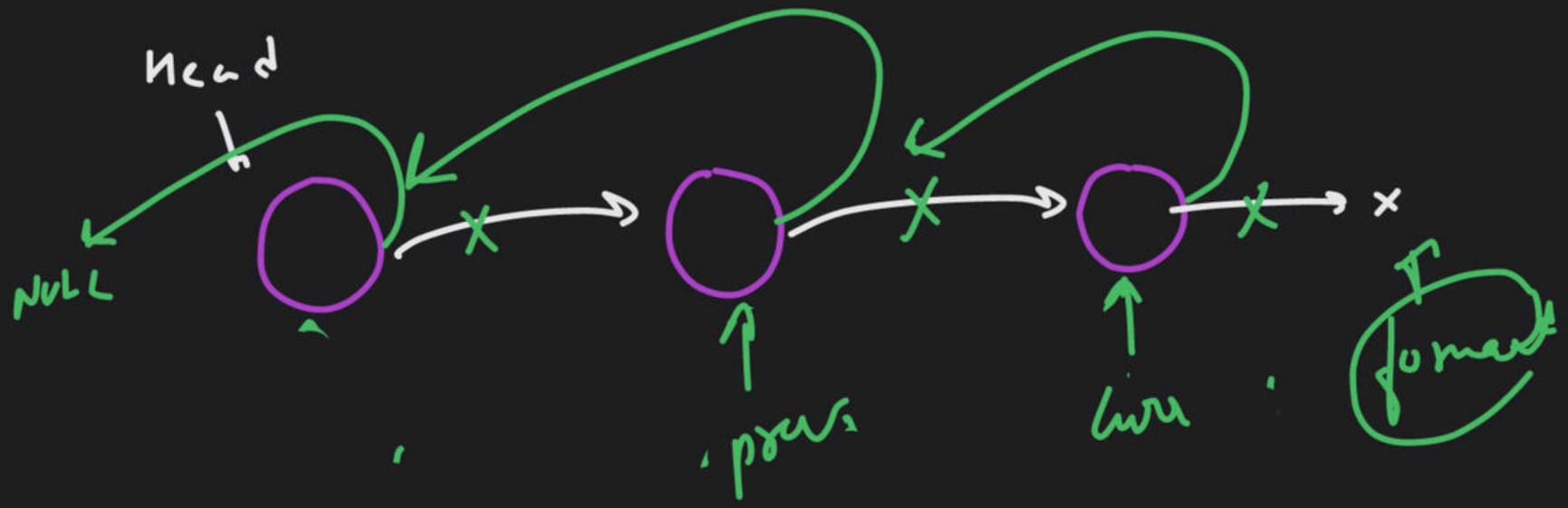

prev
curr
forward

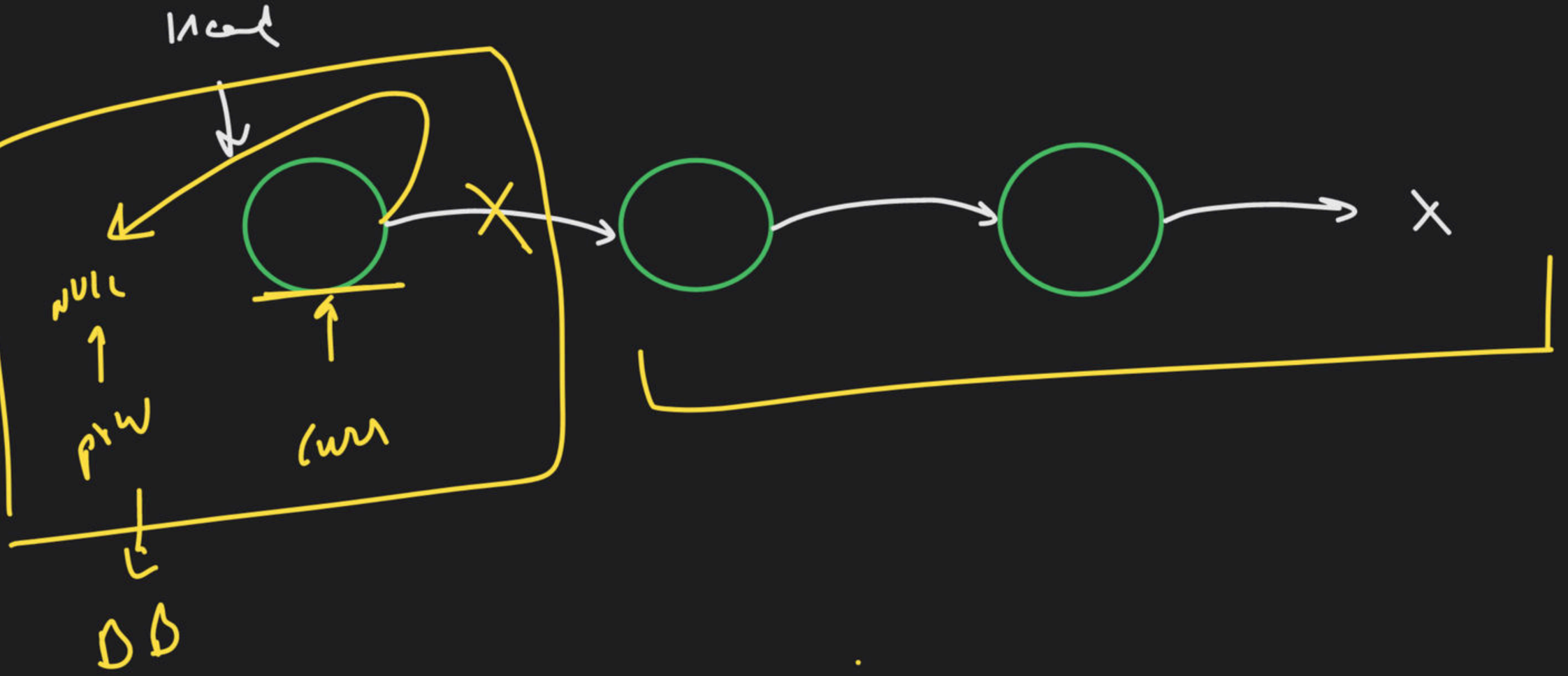
Node a : ——

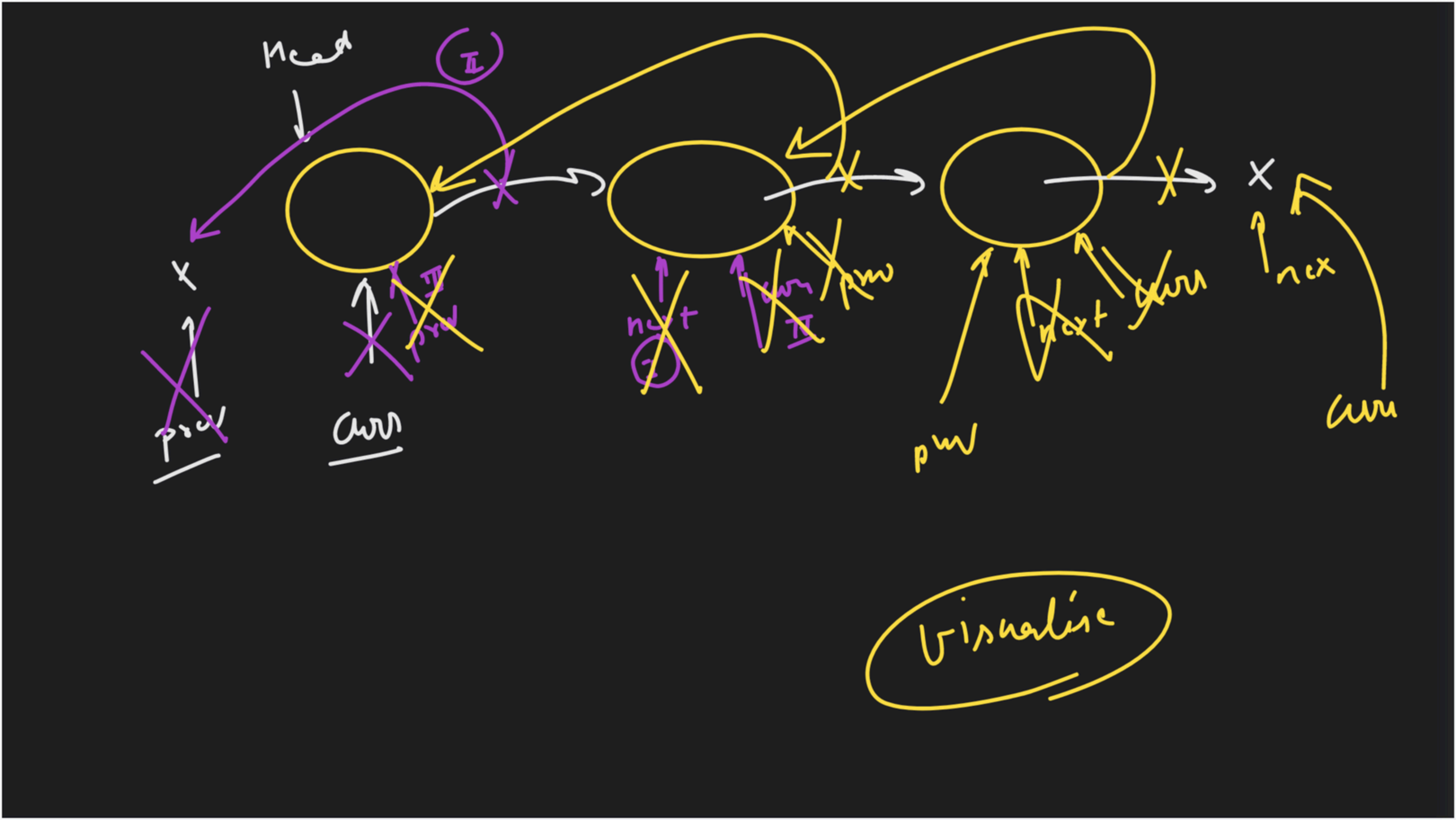


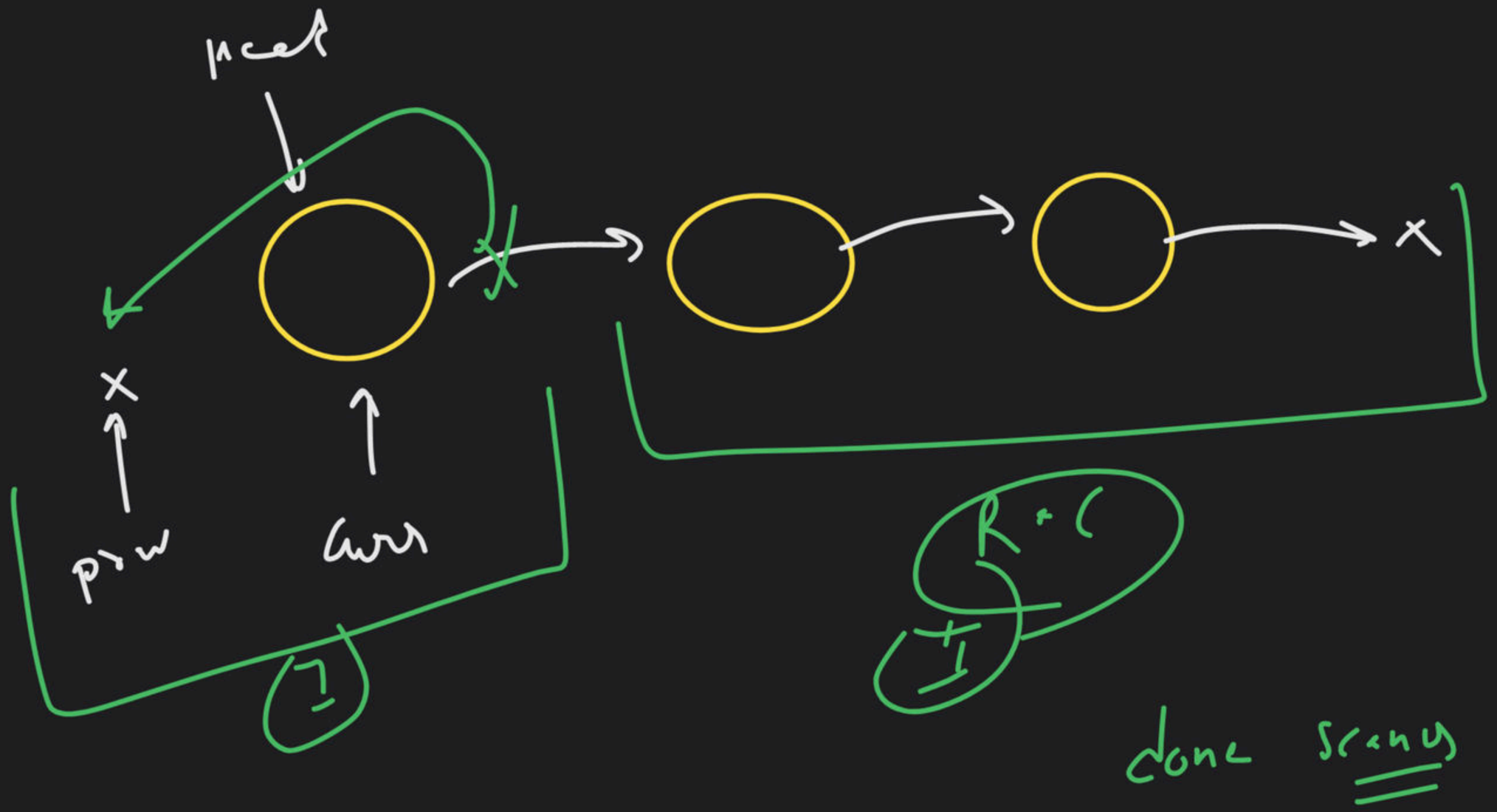
a

a


Head
NULL
x
x
x
x
prev
curr
forward

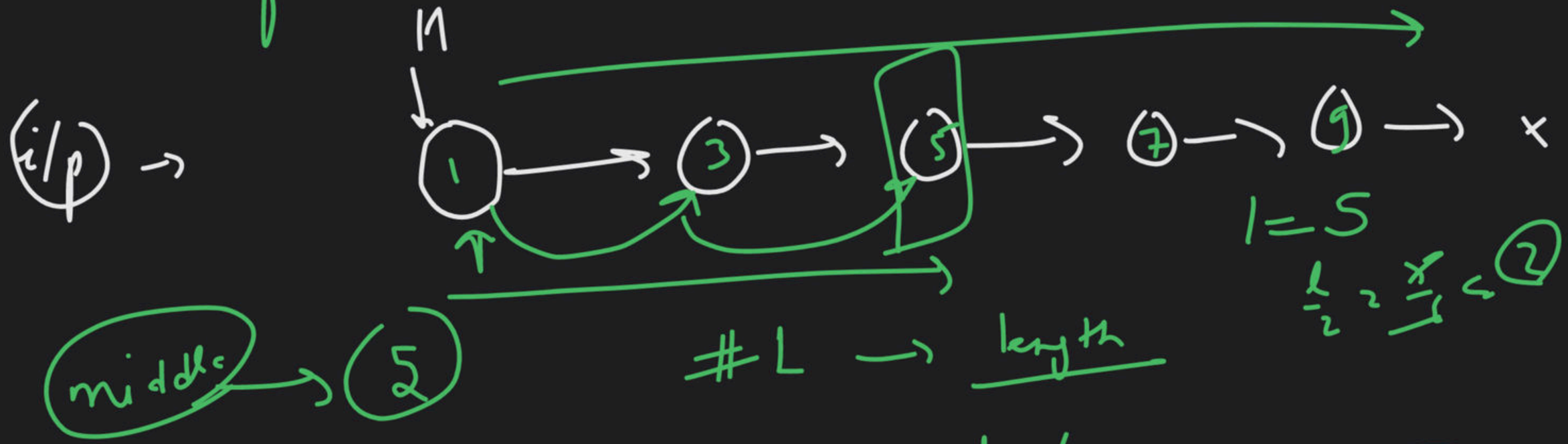

NULL
prev
curr

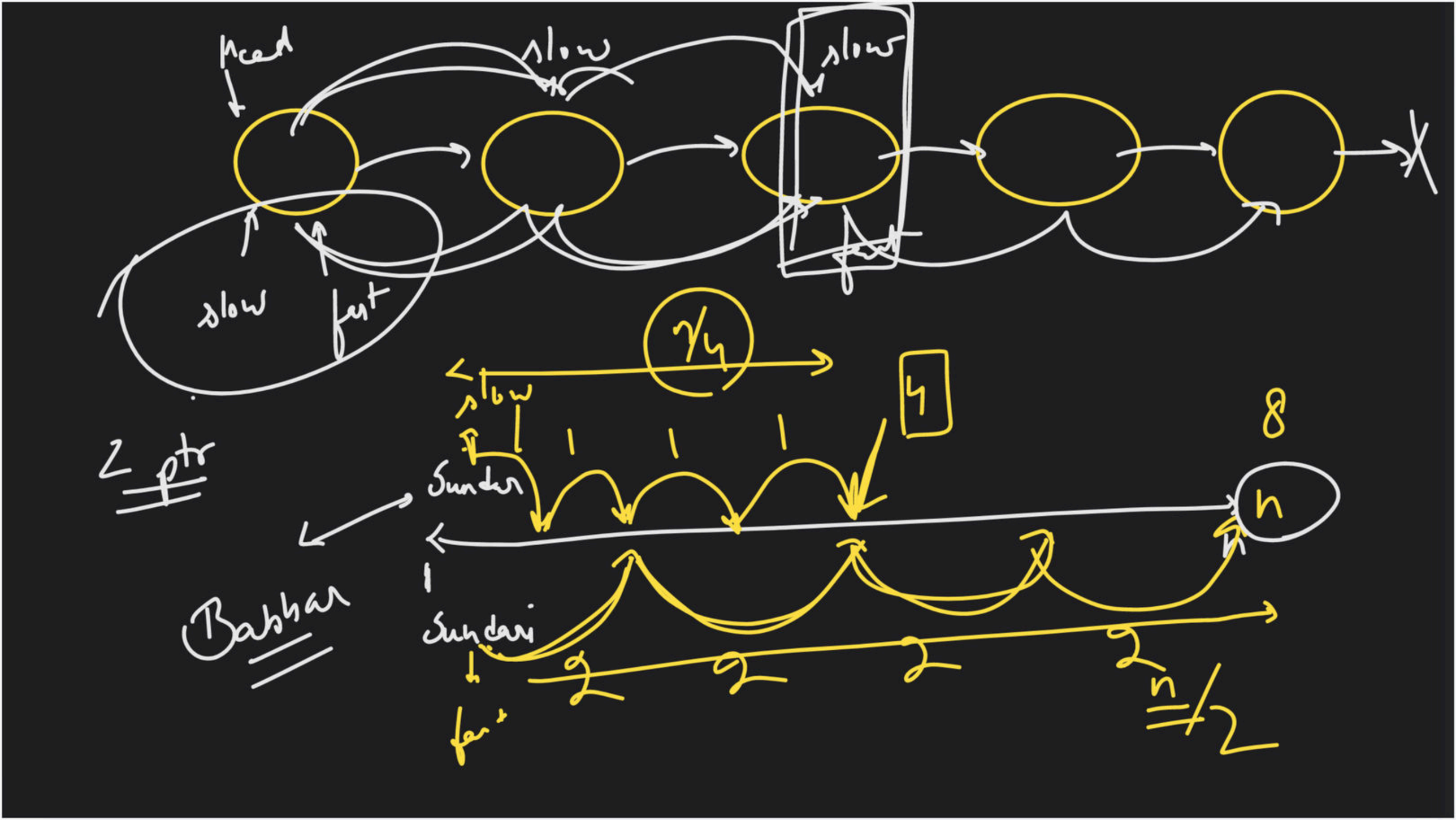

Head
x
prev
curr
next
pw
curr
next
pw
curr
x
nex
curr
visualize

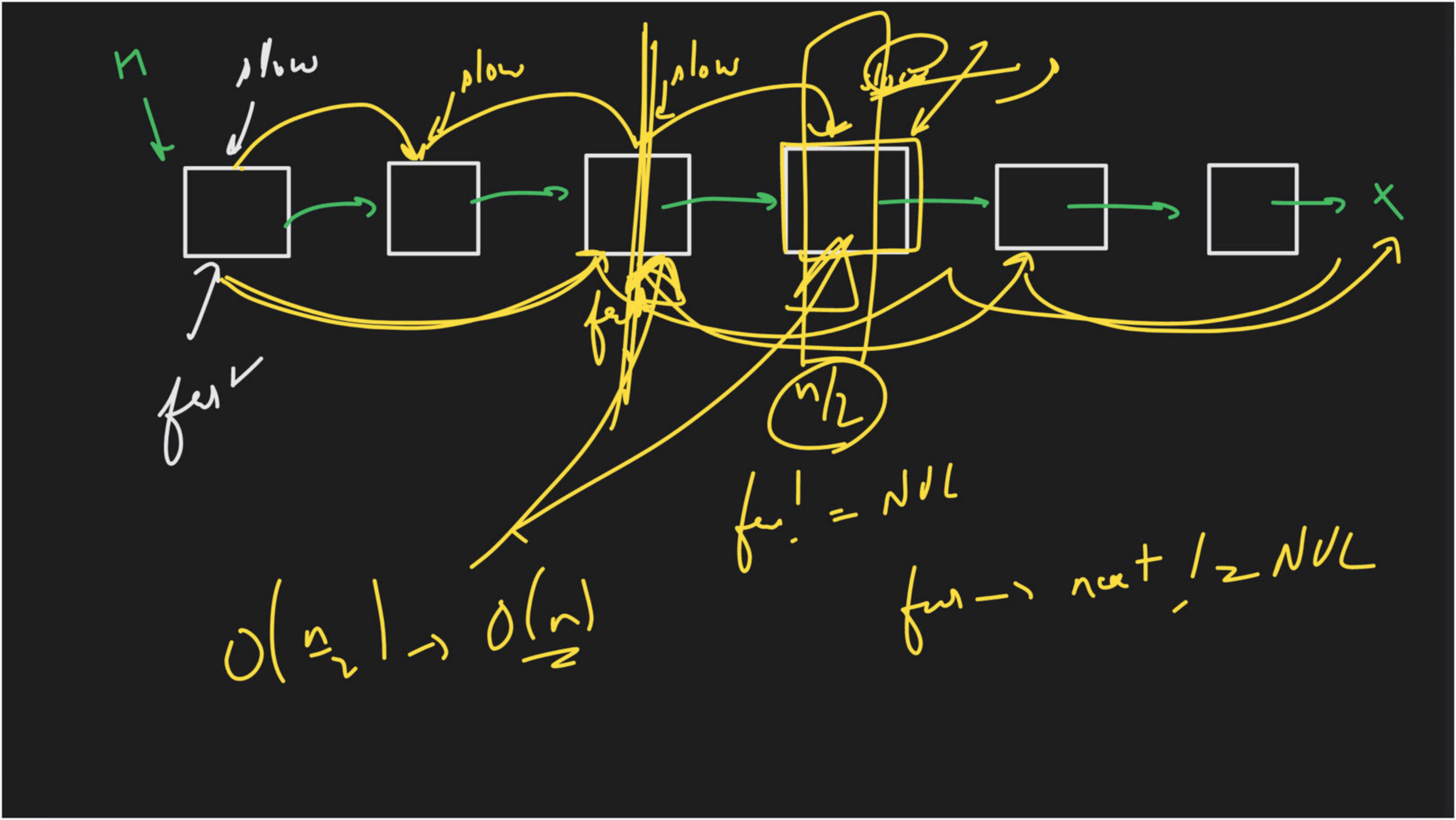

node
x
prev
curr
next
x
R + C
1
done


{
Node *
}

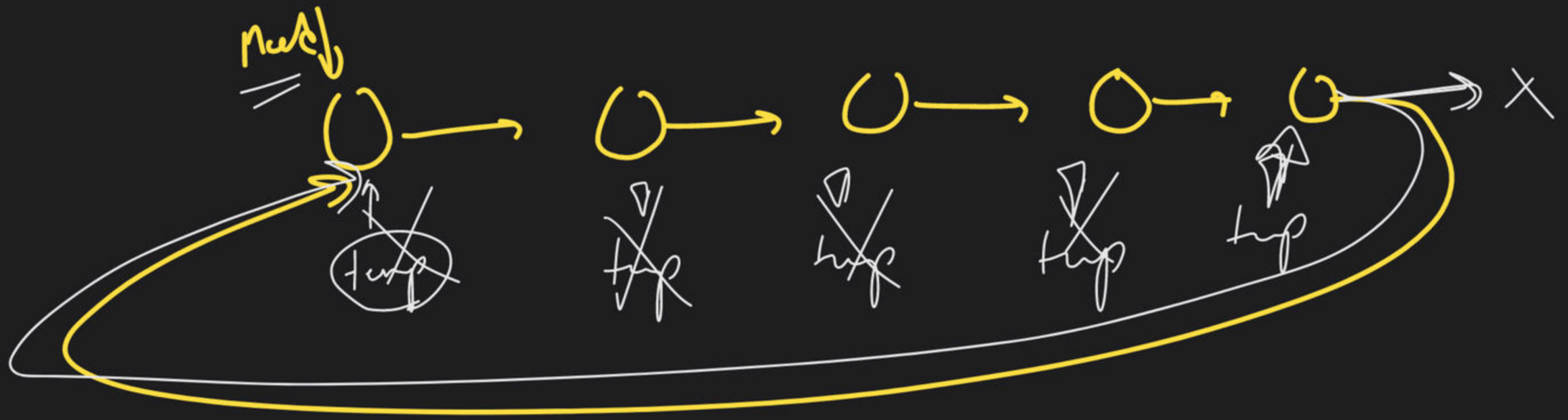
(1) Find middle of Linked List




head
slow
slow
slow
fast
2 ptr
Babbar
slow
fast
Sundar
Sundari
n/4
n
8
n/2


H
slow
slow
slow
fast
fast
n/2
fast != NULL
fast -> next != NULL
O(n/2) -> O(n)

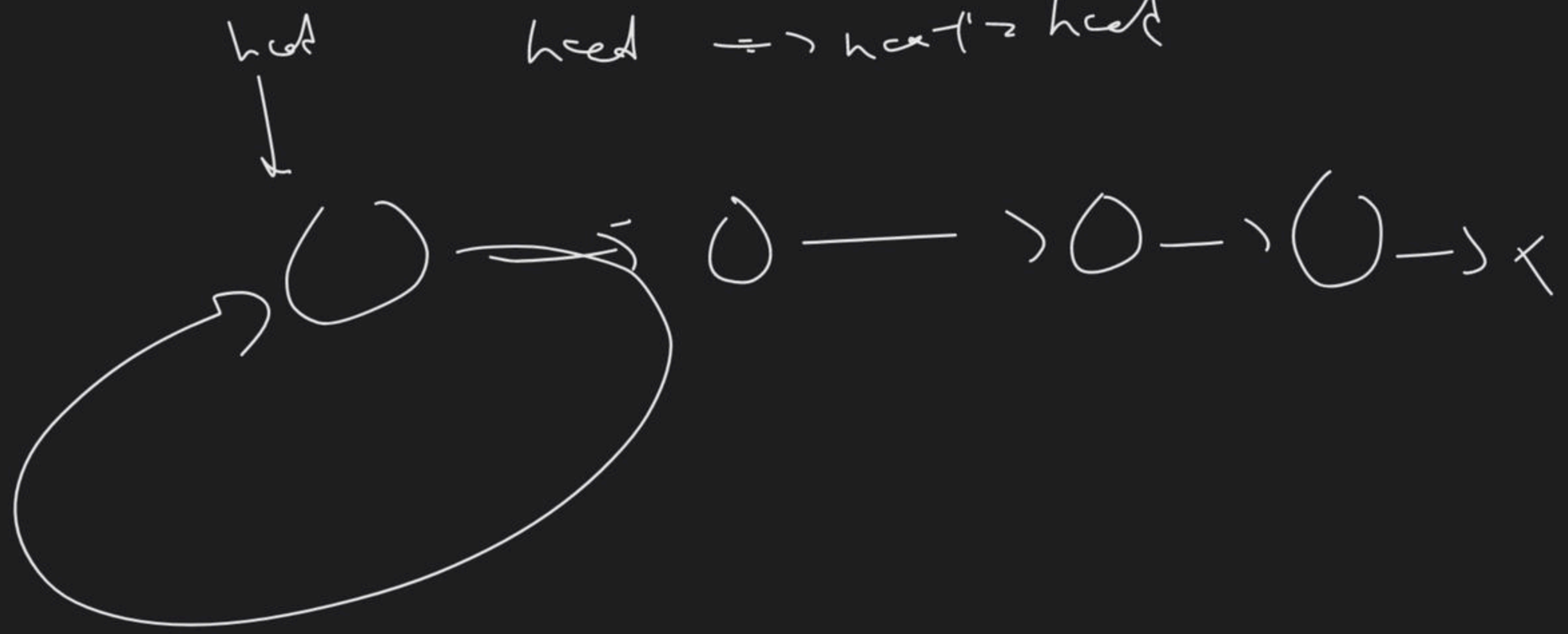
Check LL is circular or not

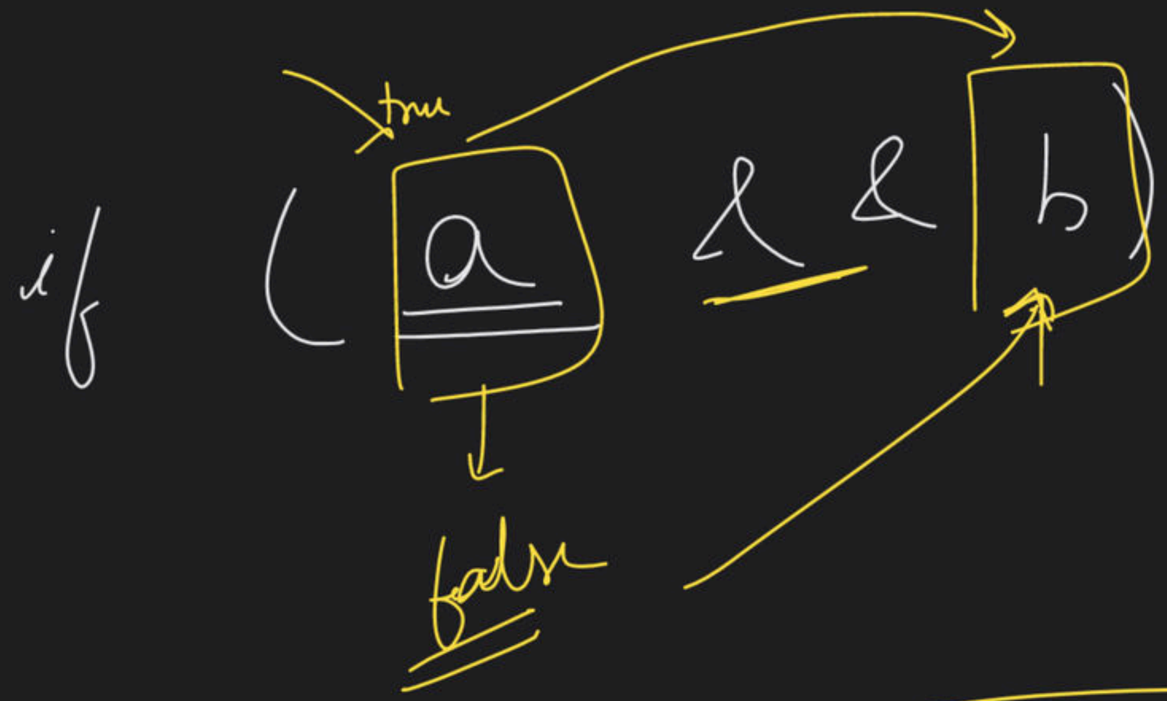


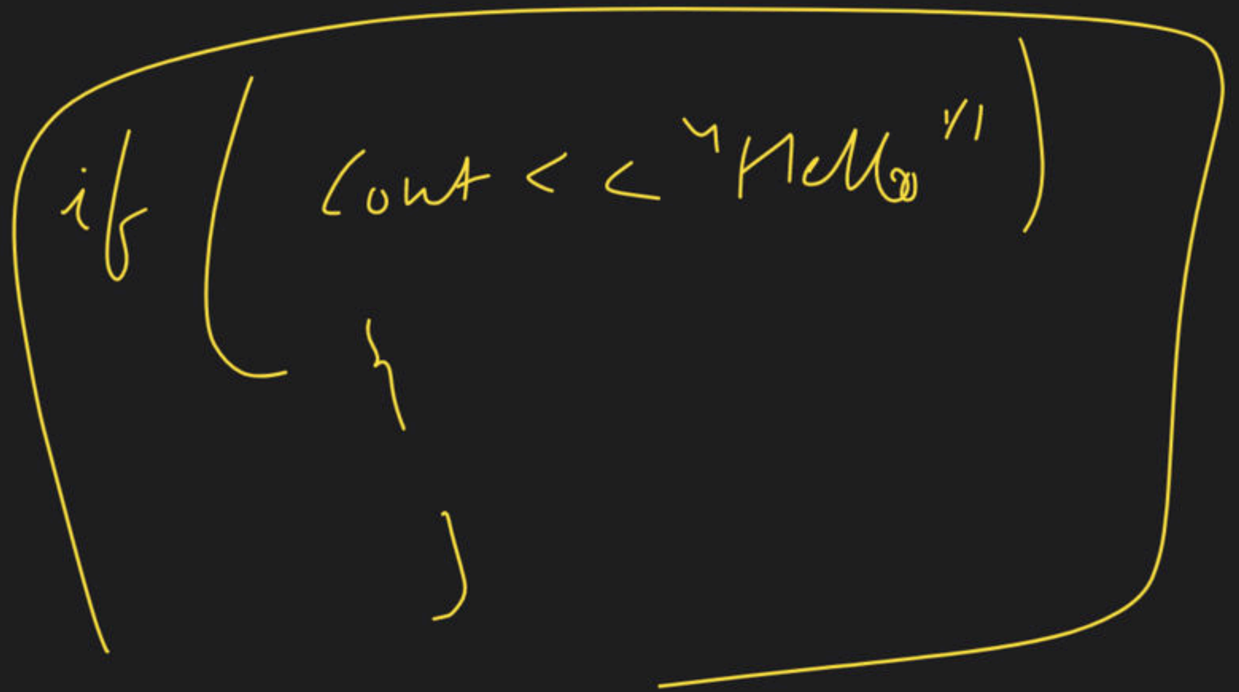

```
if (temp -> next = head)
    return true
```

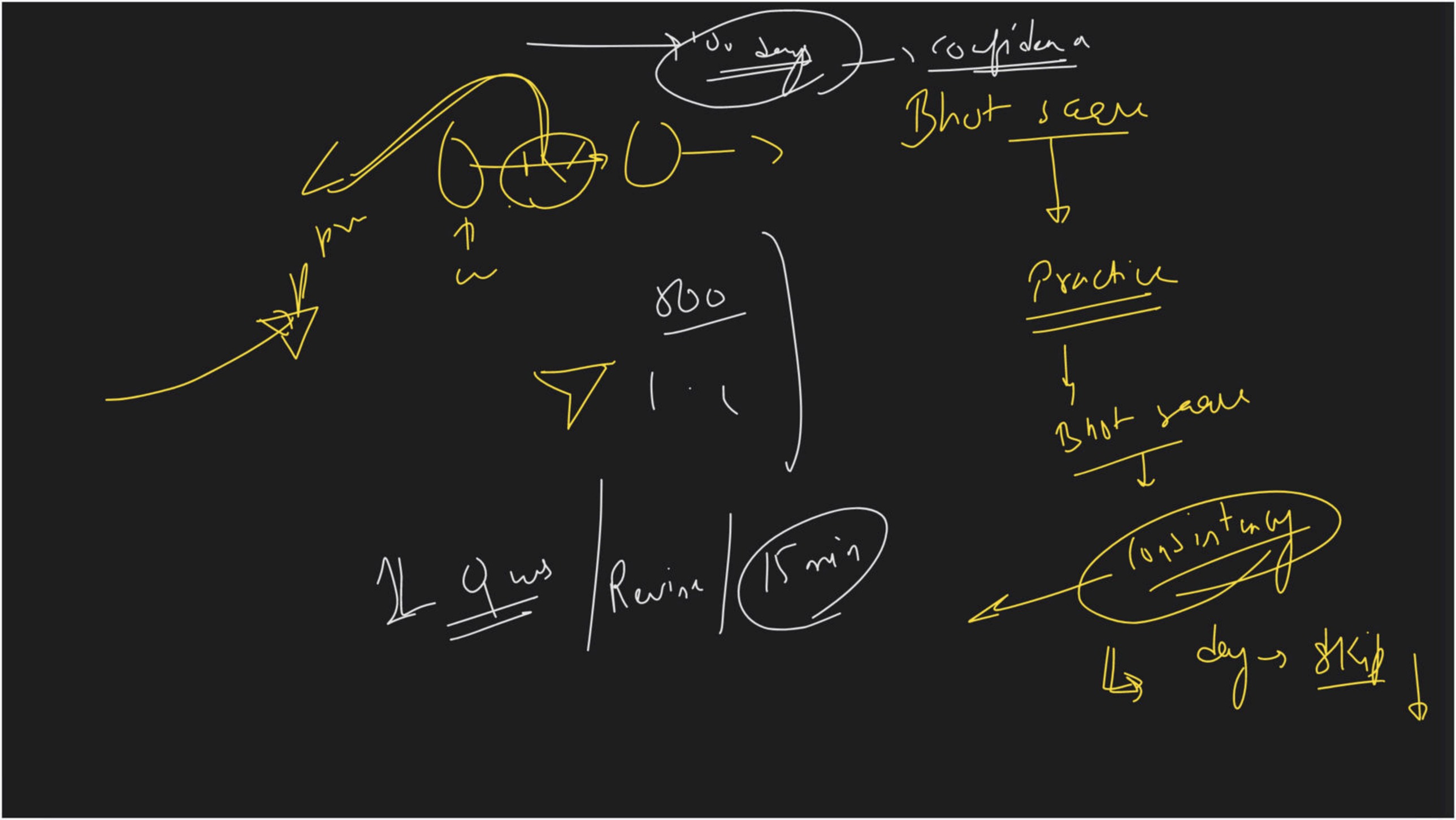
→ Circular

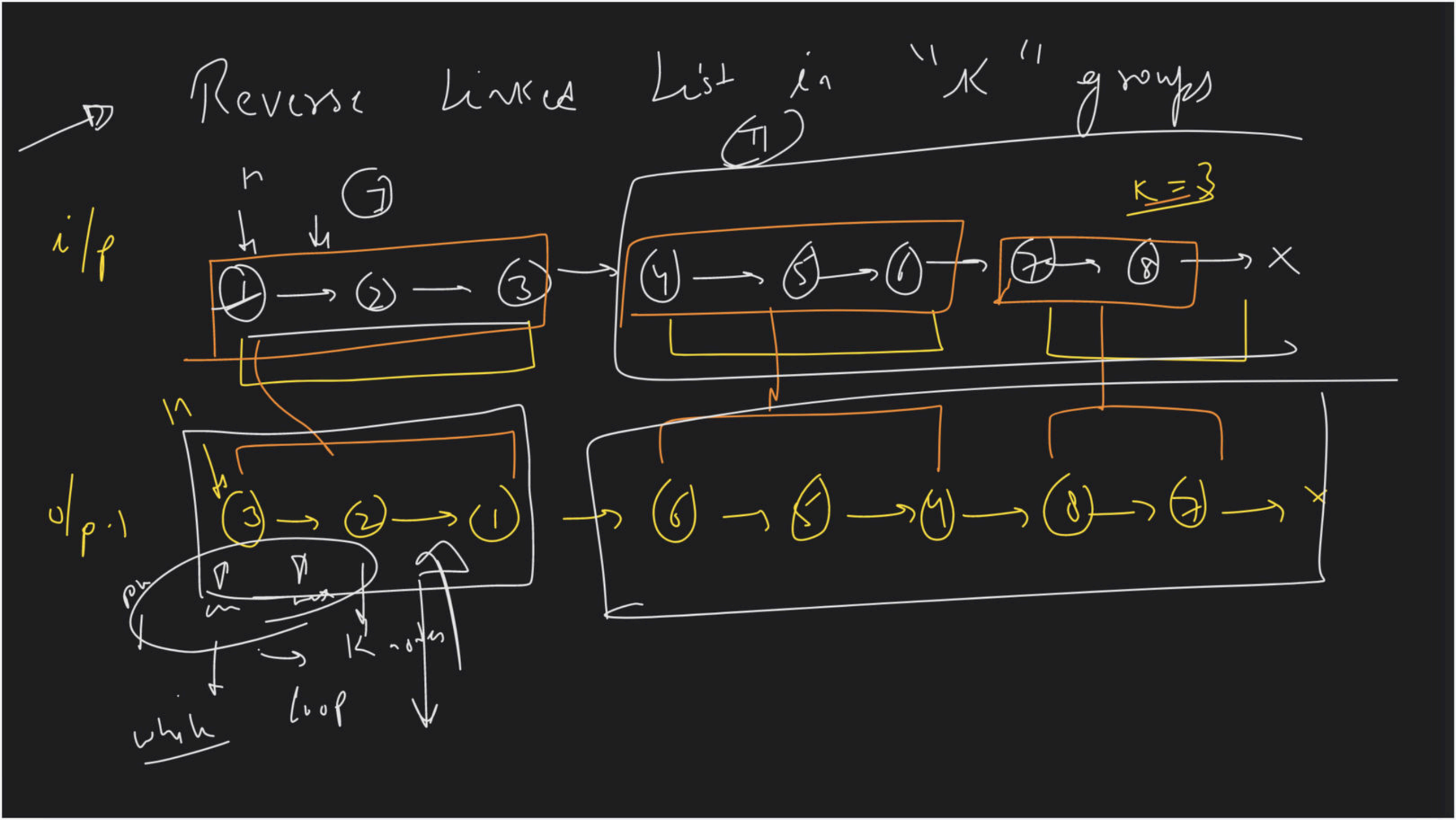
temp = temp -> next

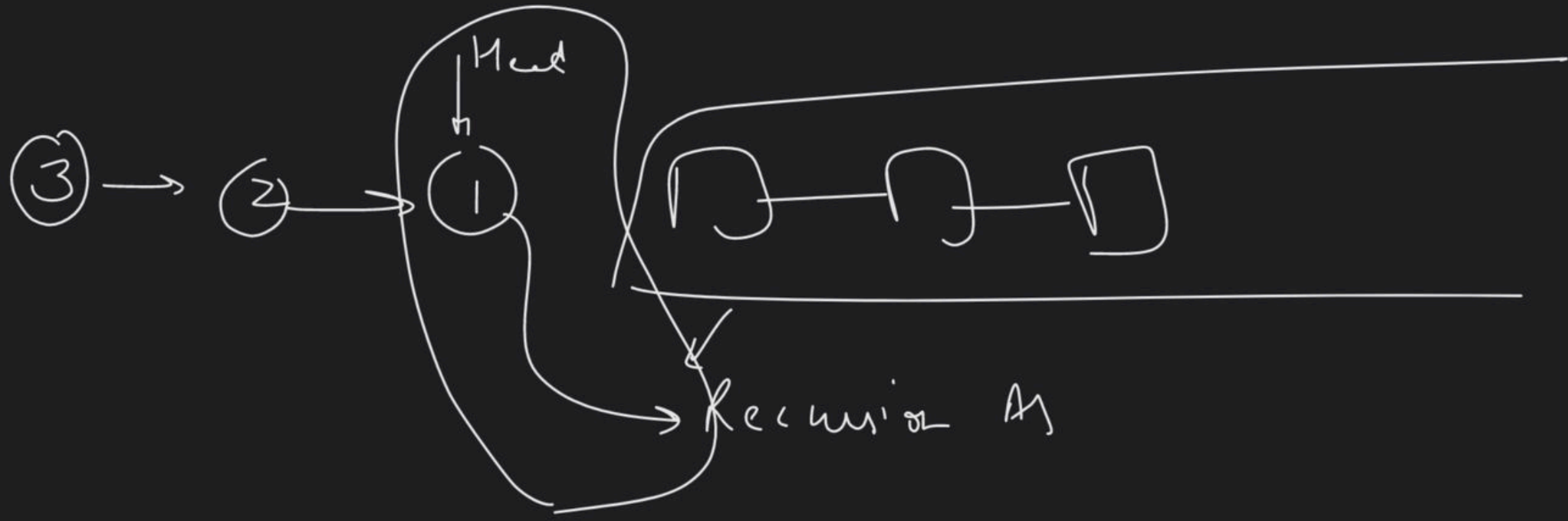

head
head => next = head

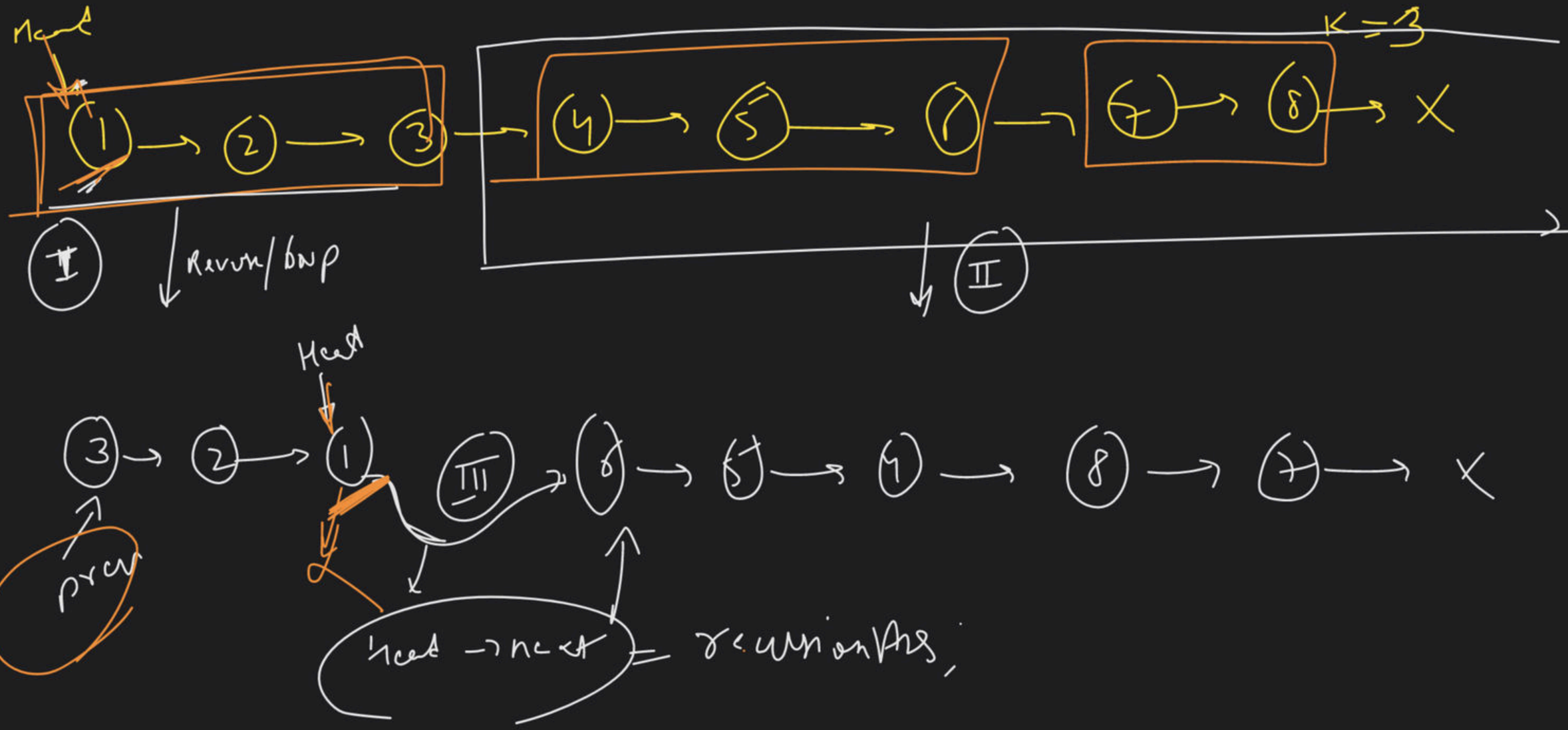

if
a
&&
b
true
false


if
cout << "Hello"


Two days
confidence
Bhut sara
Practice
Bhut sara
800
1:1
1L 9ws / Revise / 15min
Consistency
day → skip


Reverse Linked List in "K" groups
i/p
o/p
K = 3
1
2
3
4
5
6
7
8
X
K nodes
loop
while


Mid
3
2
1
Recursion


Head
K = 3
1
2
3
4
5
6
7
8
X
I
Return/Bkp
II
Head
3
2
1
III
6
5
4
8
7
X
prev
Head → next = recursionPtr;

return prev;

Algo

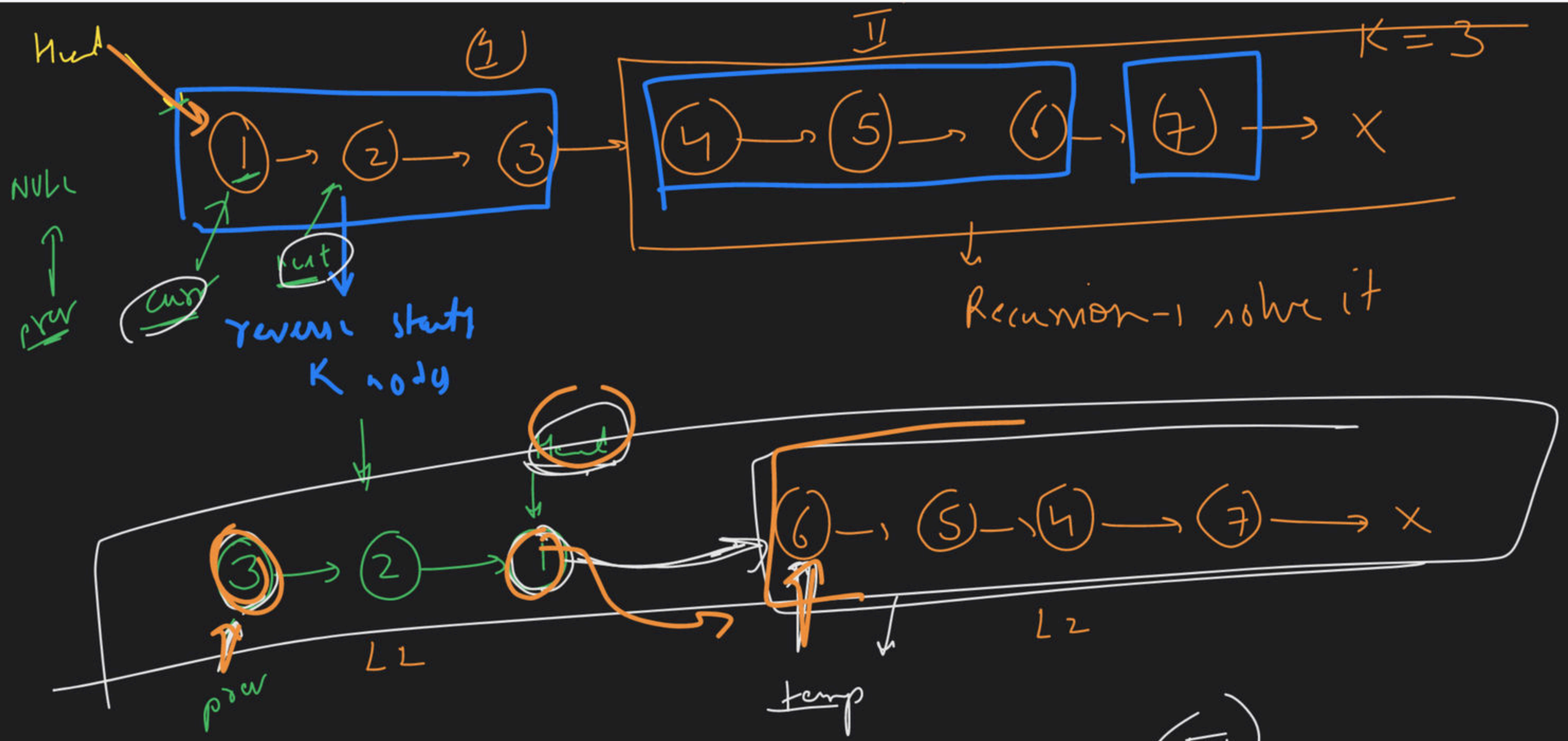
iterat

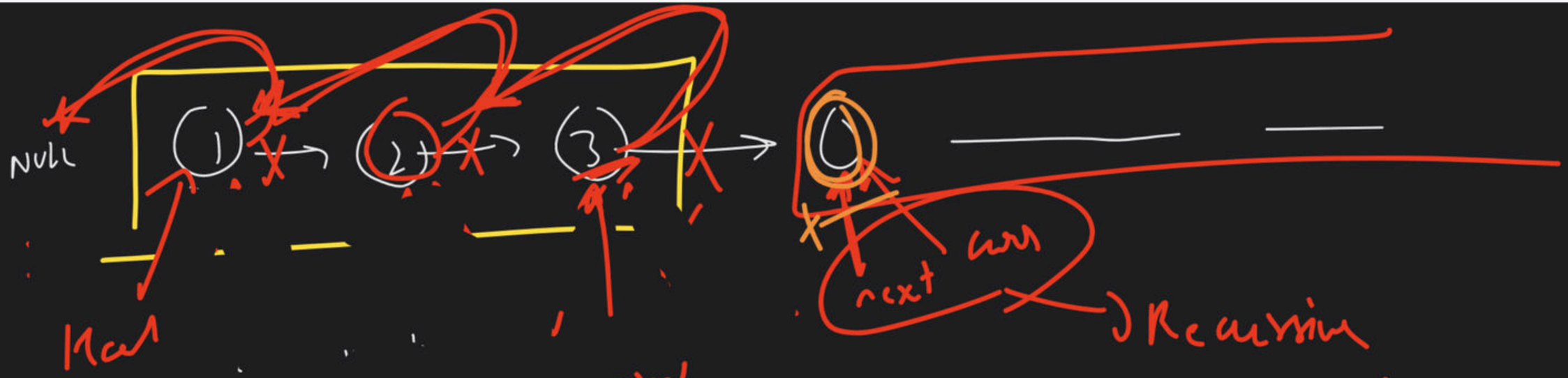
(1) Reverse first K nodes

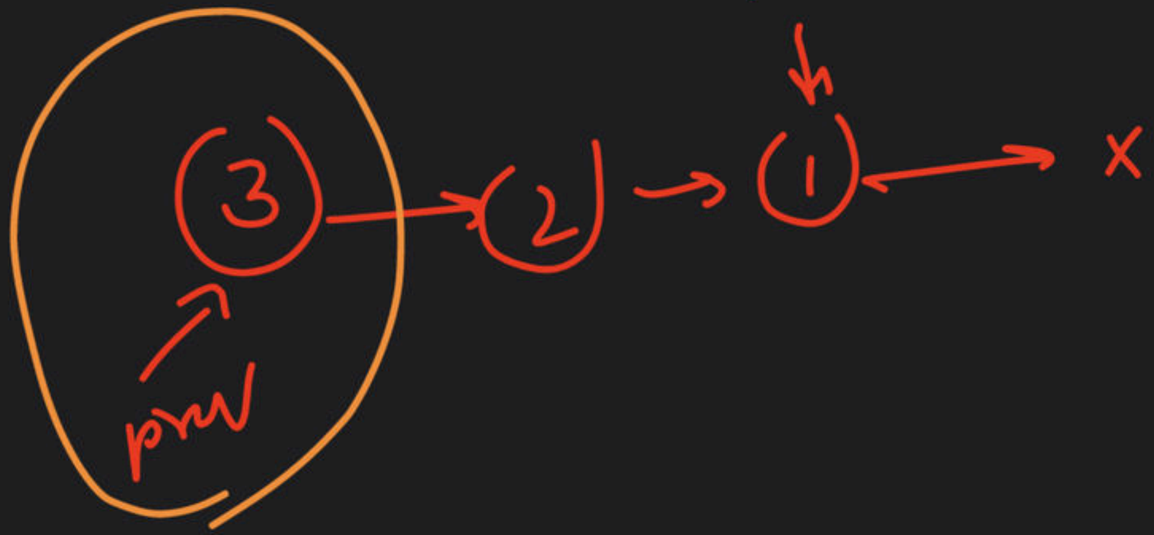
(2) Use recursion for remaining LL;

(3) Connect LL in step(1) with LL in step(2)

(4) return prev ← (head of entire List);

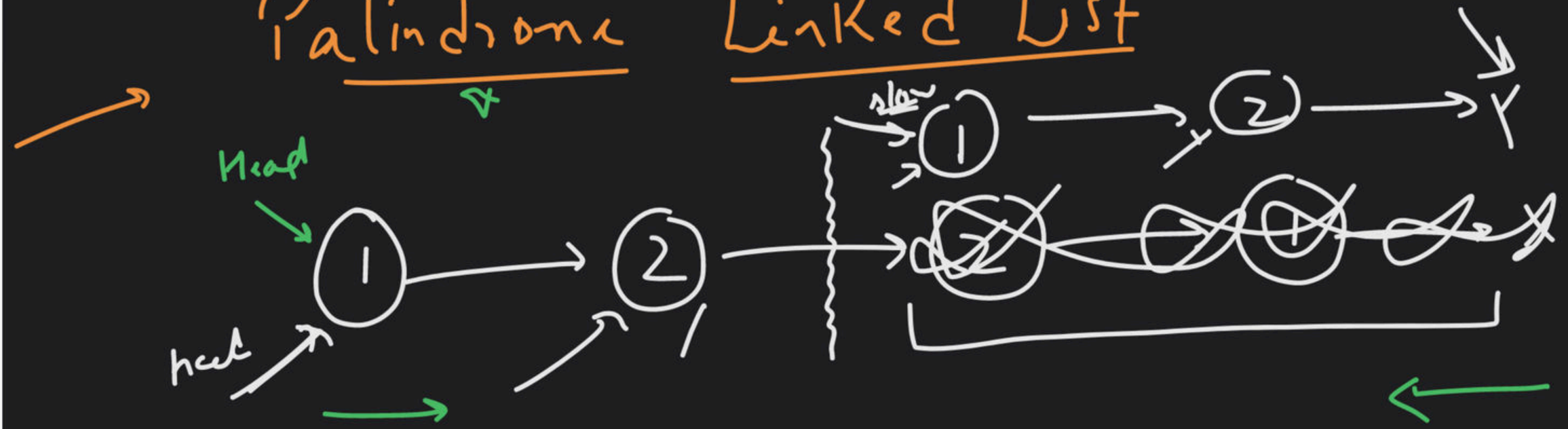

Head
I
II
K=3
1
2
3
4
5
6
7
X
NULL
prev
curr
next
reverse starts
K nody
Recursion-I solve it
temp
L2
L2
head → next = temp → III
return prev → IV





curr -> NULL

Palindrome Linked List



#1 array -> palindrome

data -> arr

#2 LL

Reverse LL - 1

-> compare

#3

Algo:-

① Break LL into 2 halves — O(n/2)

② reverse 2nd half of LL — O(n/2)

③ Compare both List — O(n/2)

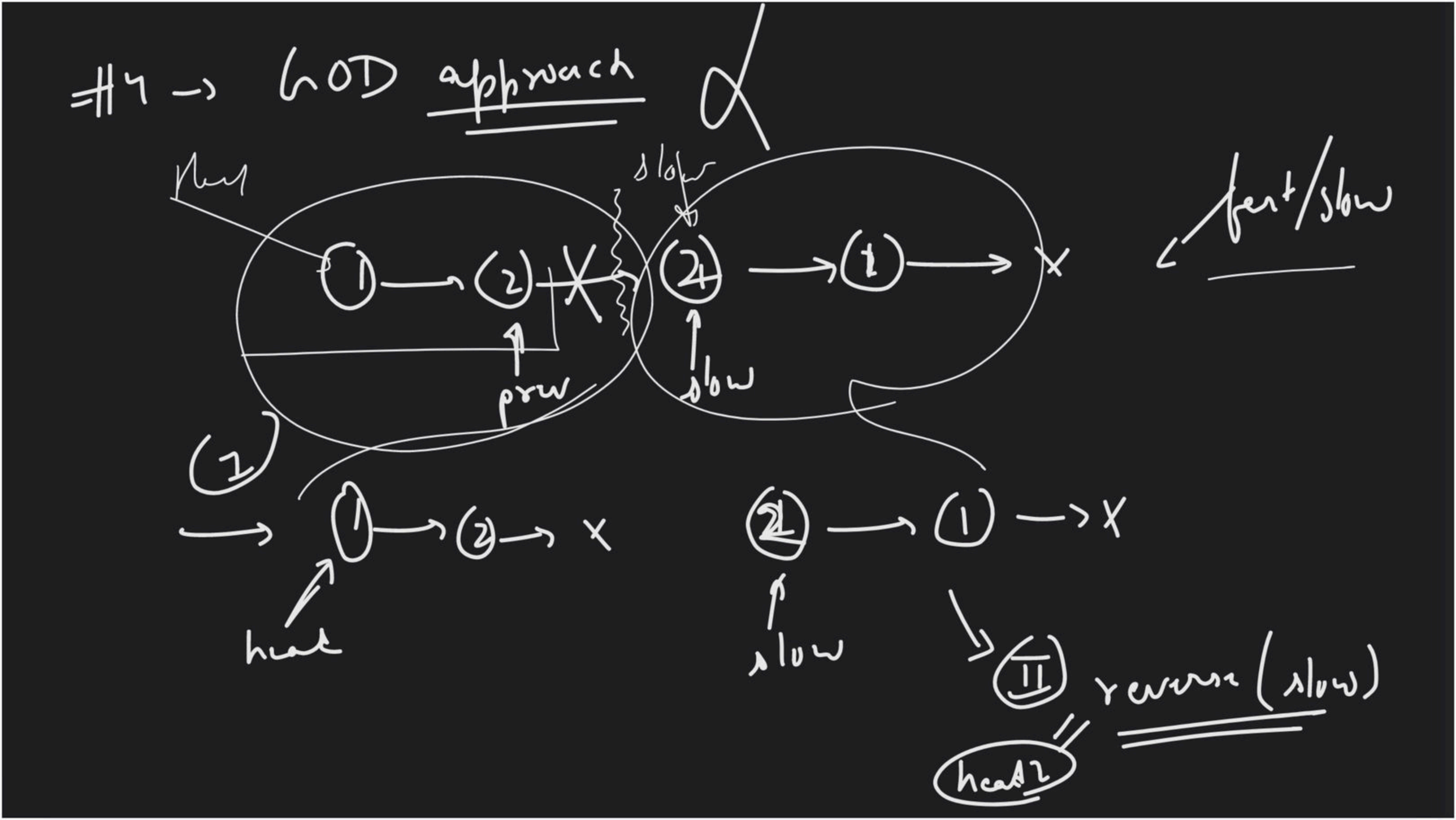
if every element is same return true

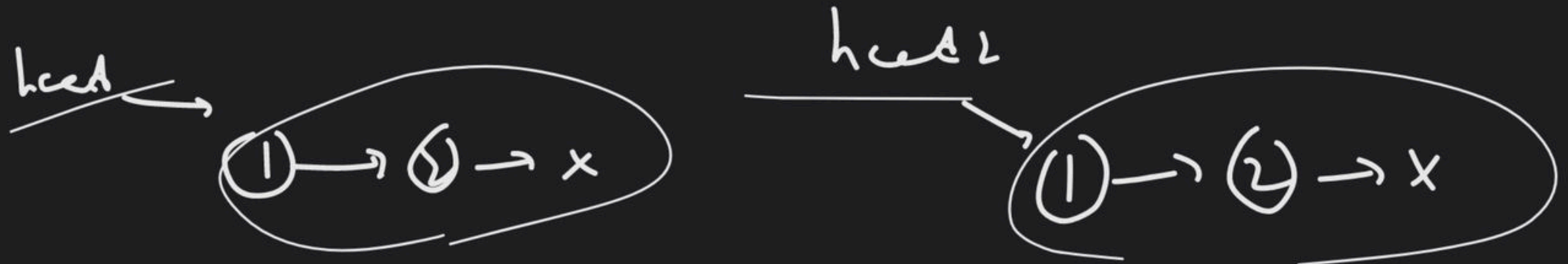
else return false

T.C ⟹ O(n)

S.C ⟹ O(1)

T.C → O(n)


#4 → GOD approach
fast
slow
1
2
3
1
X
fast/slow
prev
slow
2
1
2
X
head
2
1
X
slow
II
reverse (slow)
head2



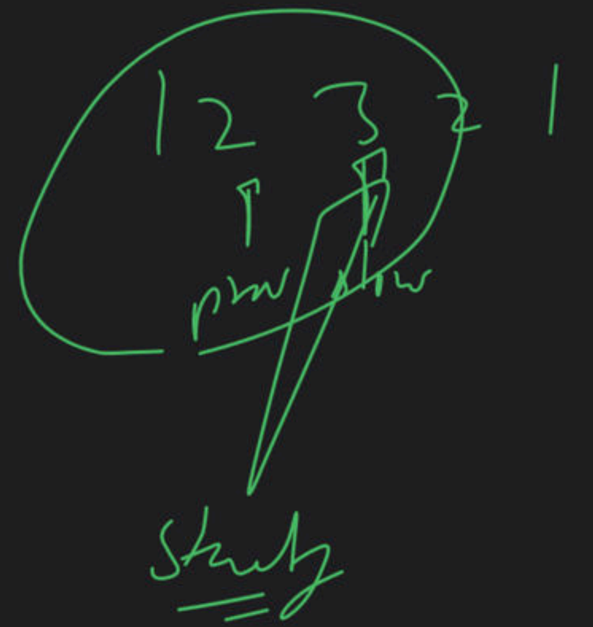

```
while ( temp1 != NULL && temp2 != NULL)
{
        if ( temp1->data != temp2->data)
            return false;
        temp1 = temp1->next
        temp2 = temp2->next
}
return true;
```

if (temp1 == NULL && temp2 == NULL)

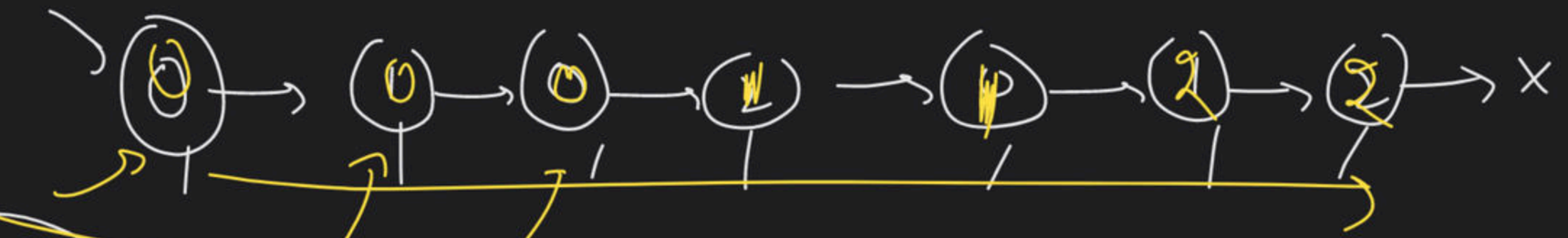
return true;

else

return false;



① Sort 0's, 1's, 2's in Linked List



implement

↓

Make

↓

edge case

#1

count

zero = 3 % 0 =

one = 2 % 0

two = 2 % 0

→ O(n)

traverse → O(n)

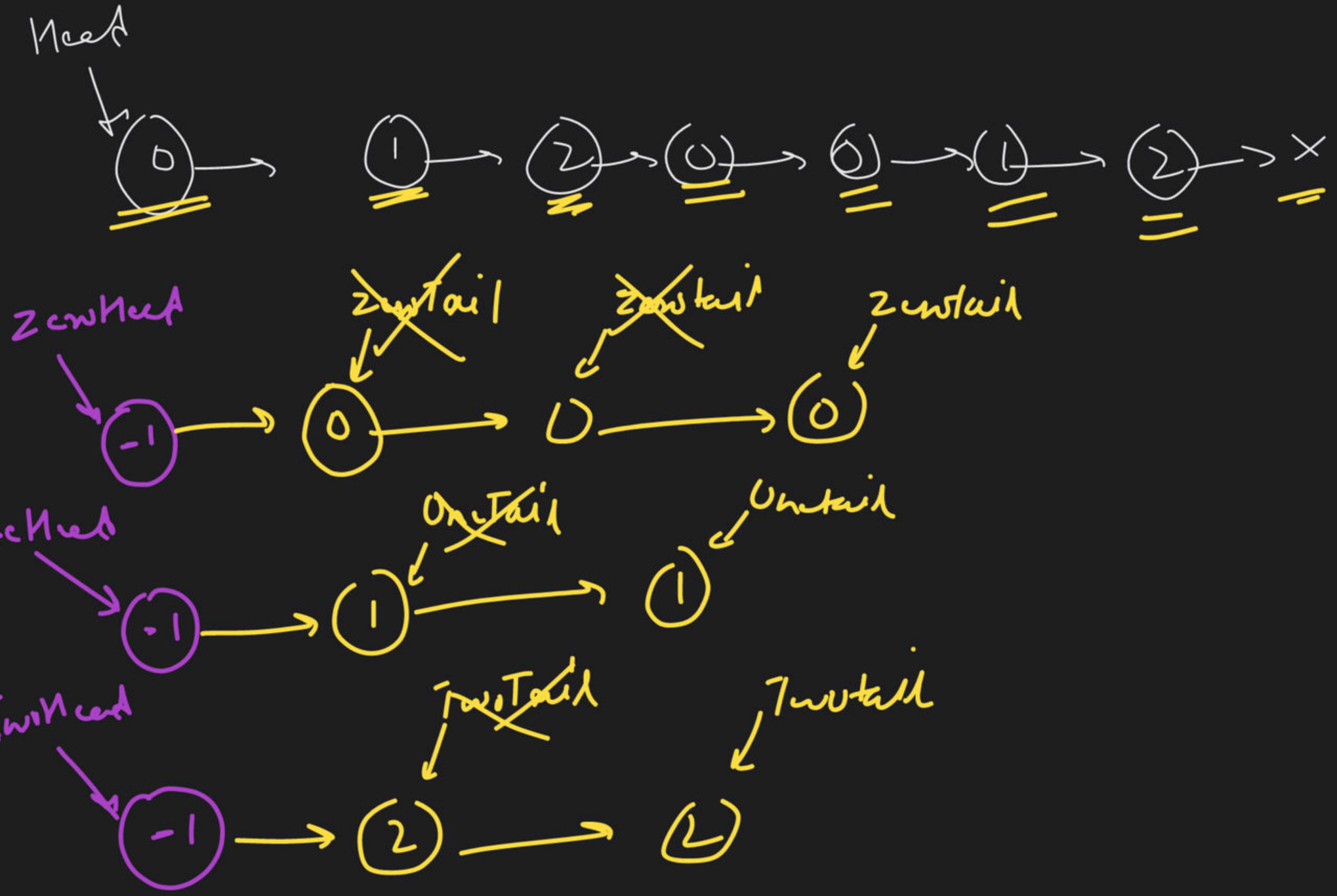
+

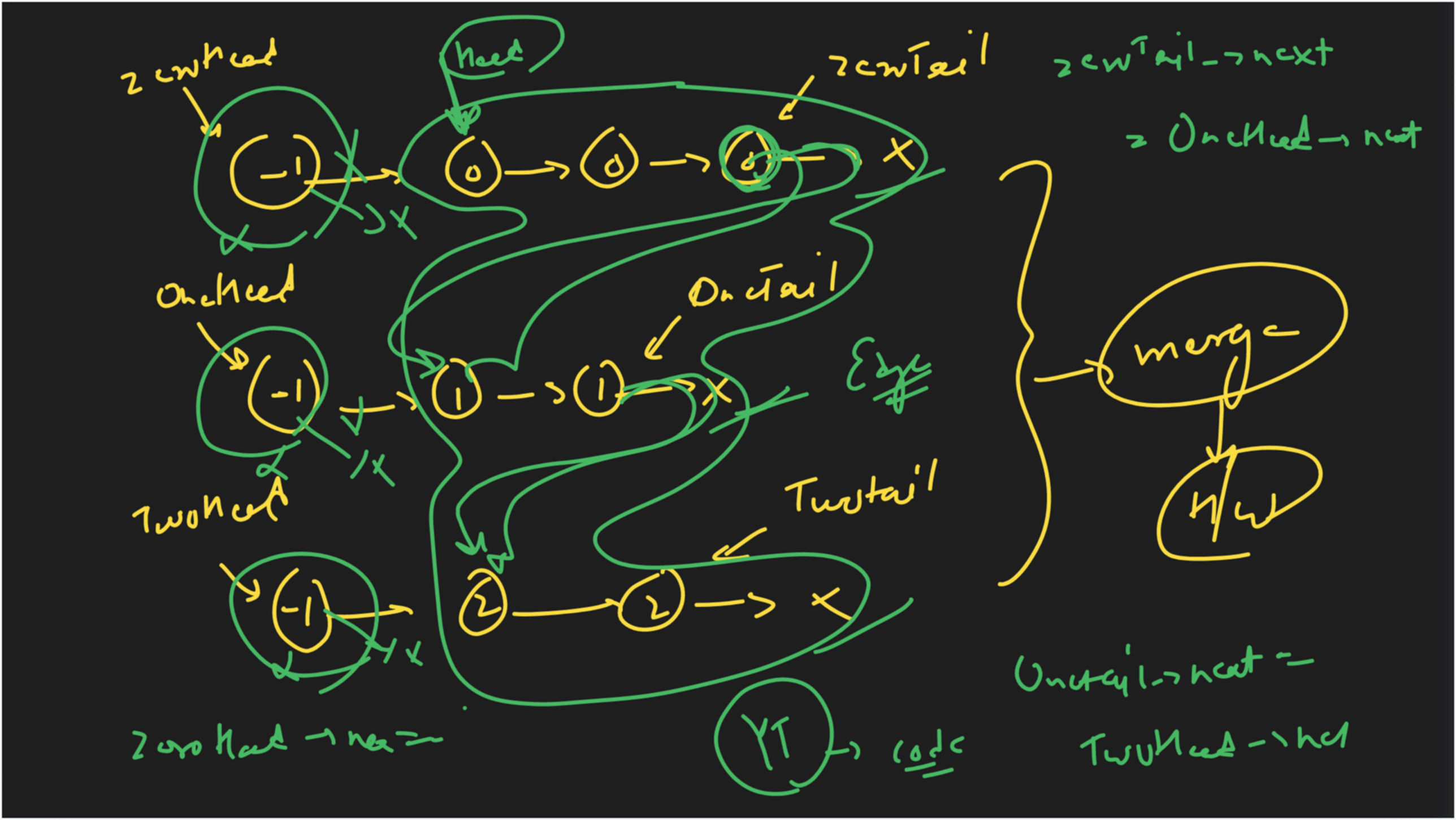
→ O(n)

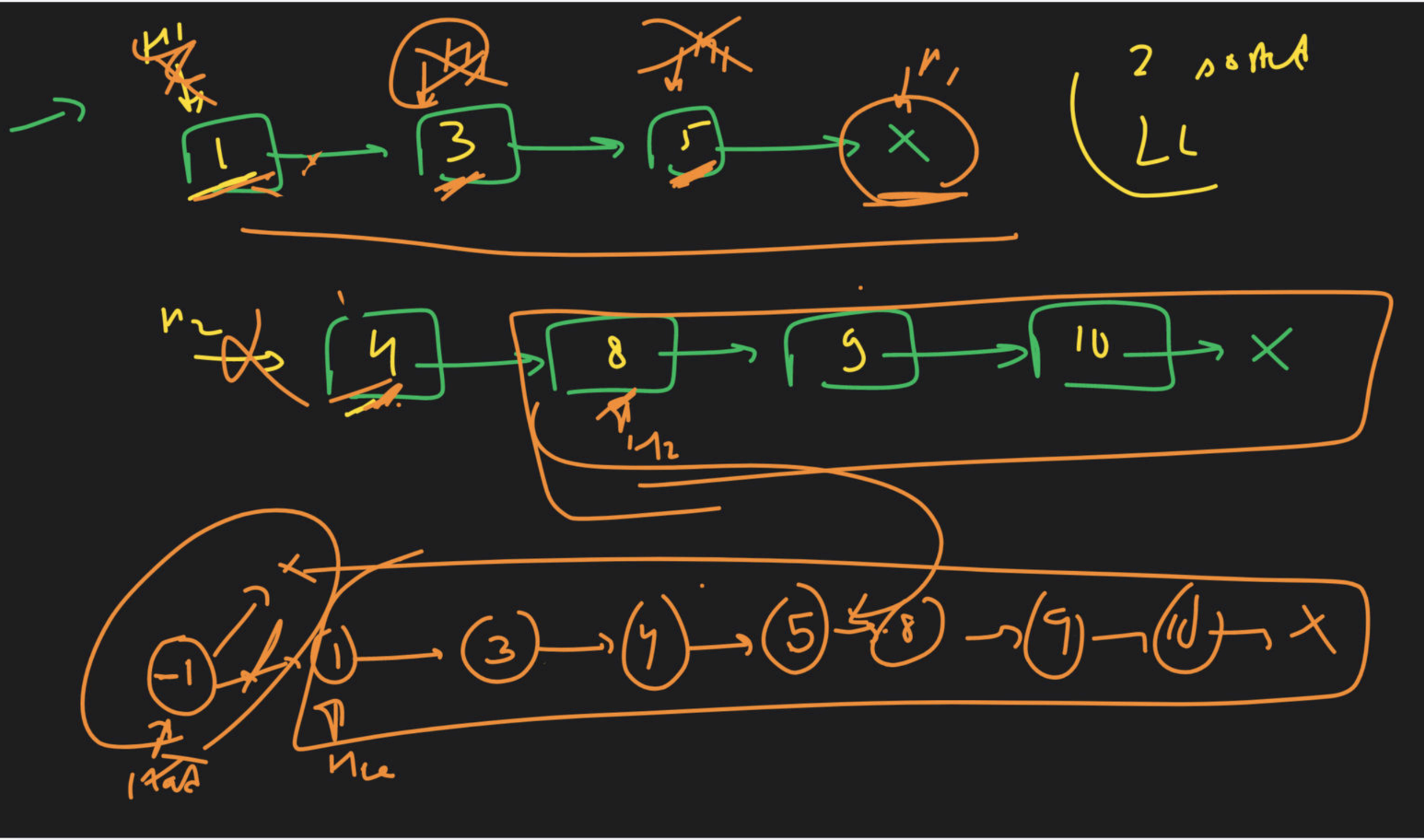
value replace

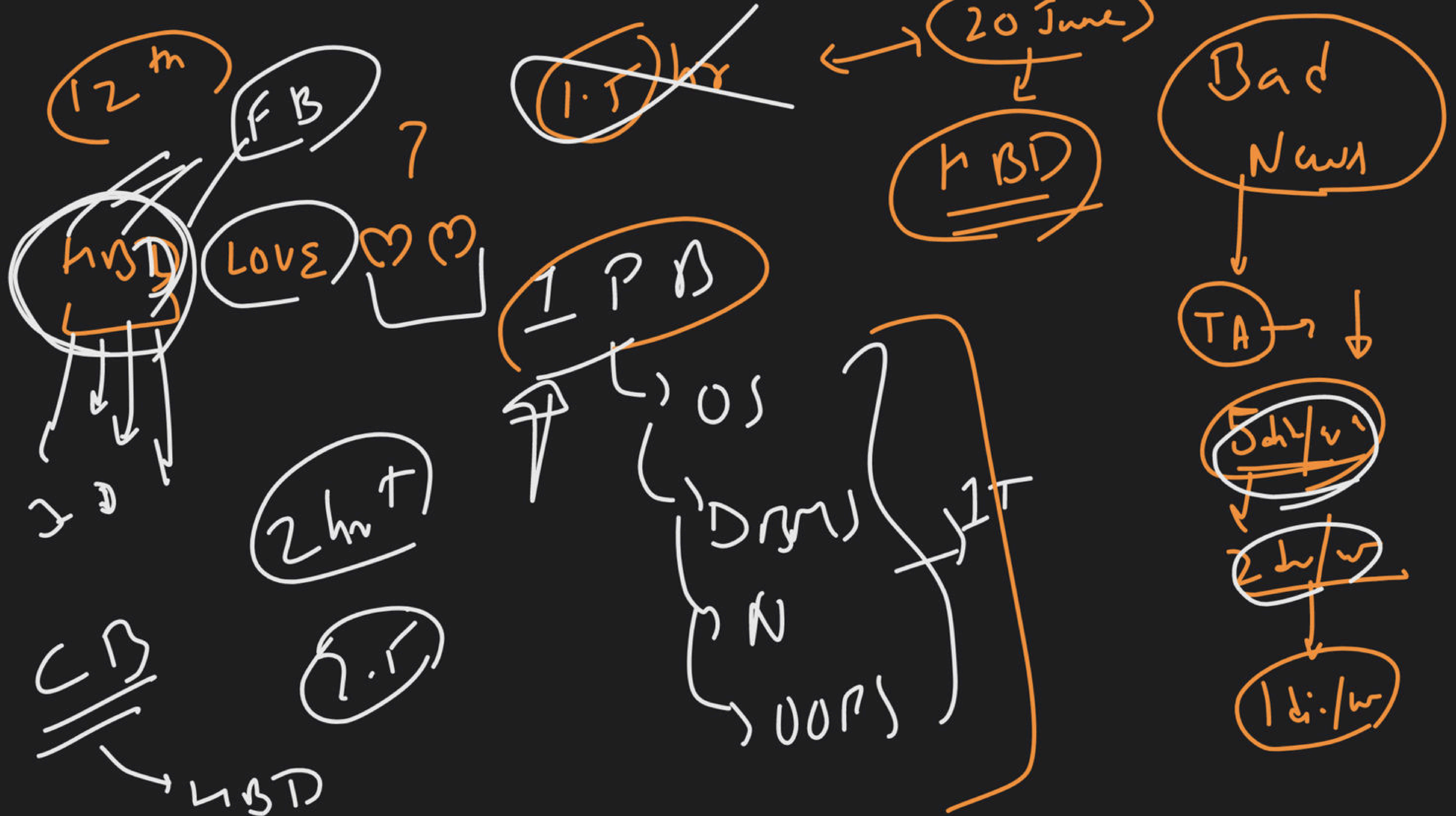
↓

links change krke answer do


Head
0
1
2
0
0
1
2
X
zeroHead
zeroTail
zeroTail
zeroTail
-1
0
0
0
oneHead
oneTail
oneTail
-1
1
1
twoHead
twoTail
twoTail
-1
2
2


zeroHead
Head
zeroTail
zeroTail->next
=OneHead->next
-1
0
0
0
X
OneHead
OneTail
-1
1
1
X
Edge
TwoHead
TwoTail
-1
2
2
X
merge
YT
code
zeroHead->next=
OneTail->next=
TwoHead->hel


1
3
5
X
2 sorted LL
4
8
9
10
X
-1
1
3
4
5
8
9
10
X


12th
FB
HBD
LOVE
20 June
HBD
Bad News
IPB
OS
DBMS
N
OOPS
IT
TA
CB
HBD