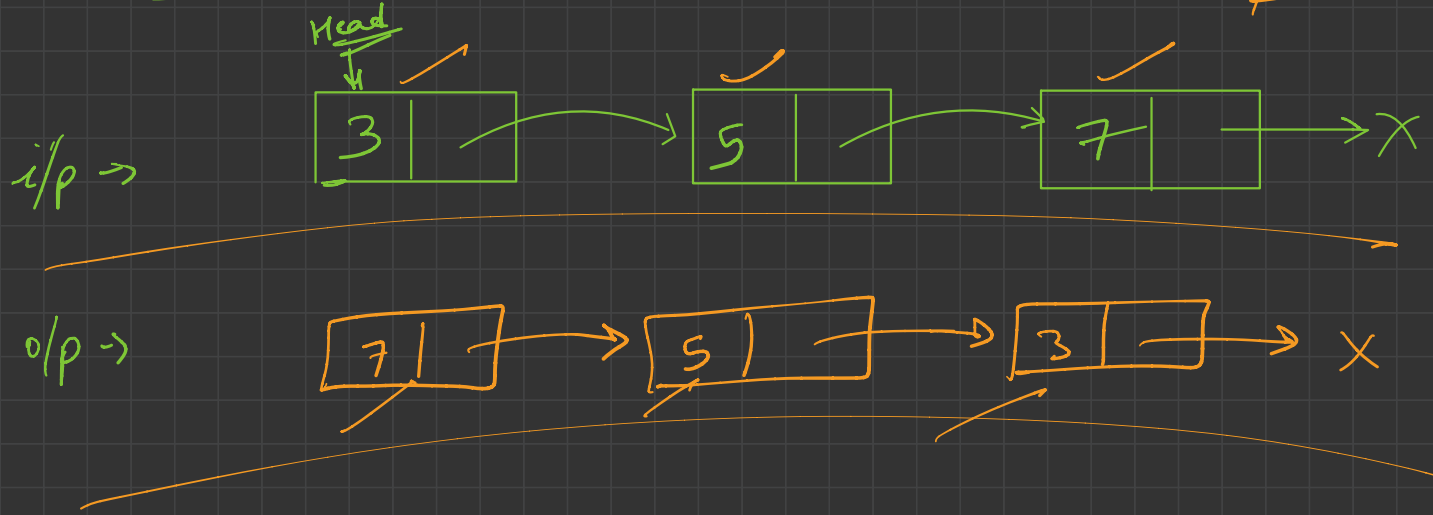
Linked List Questions :-

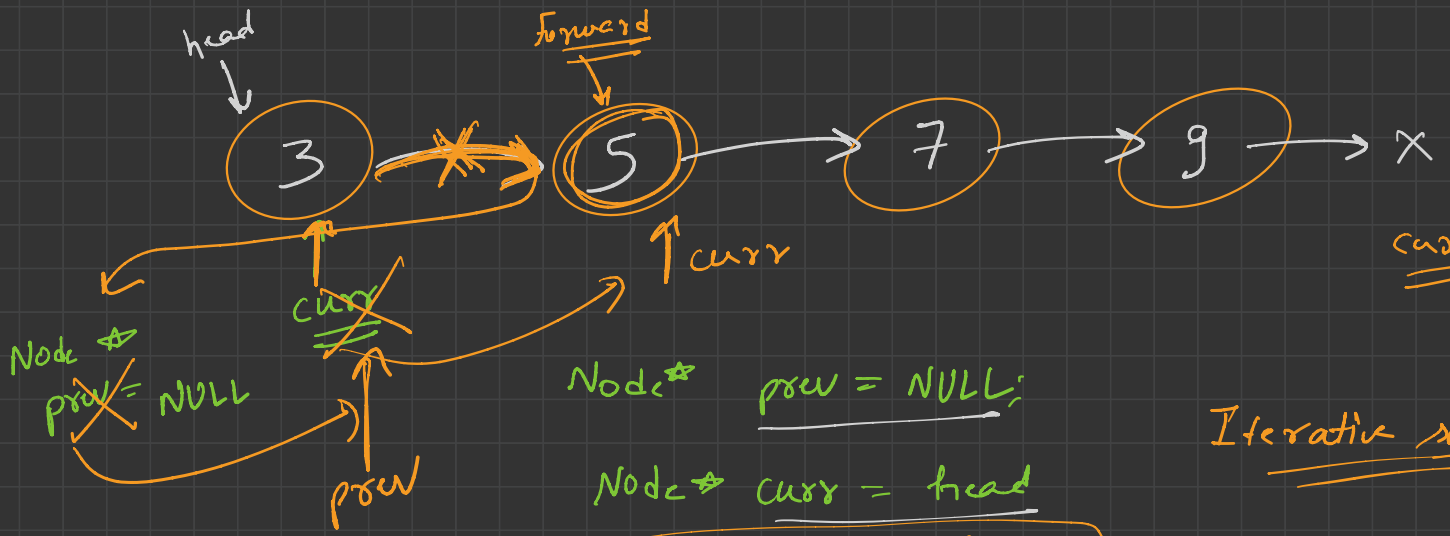
① Reverse a linked list :-

arr → [7|4|3|2]

rev → [2|3|4|7]



Approach :-



Node* prev = NULL;

Node* curr = head

Iterative solve

```
while (curr != NULL)
{ forward = curr -> next

  curr -> next = prev

  prev = curr;
  curr = forward;
```

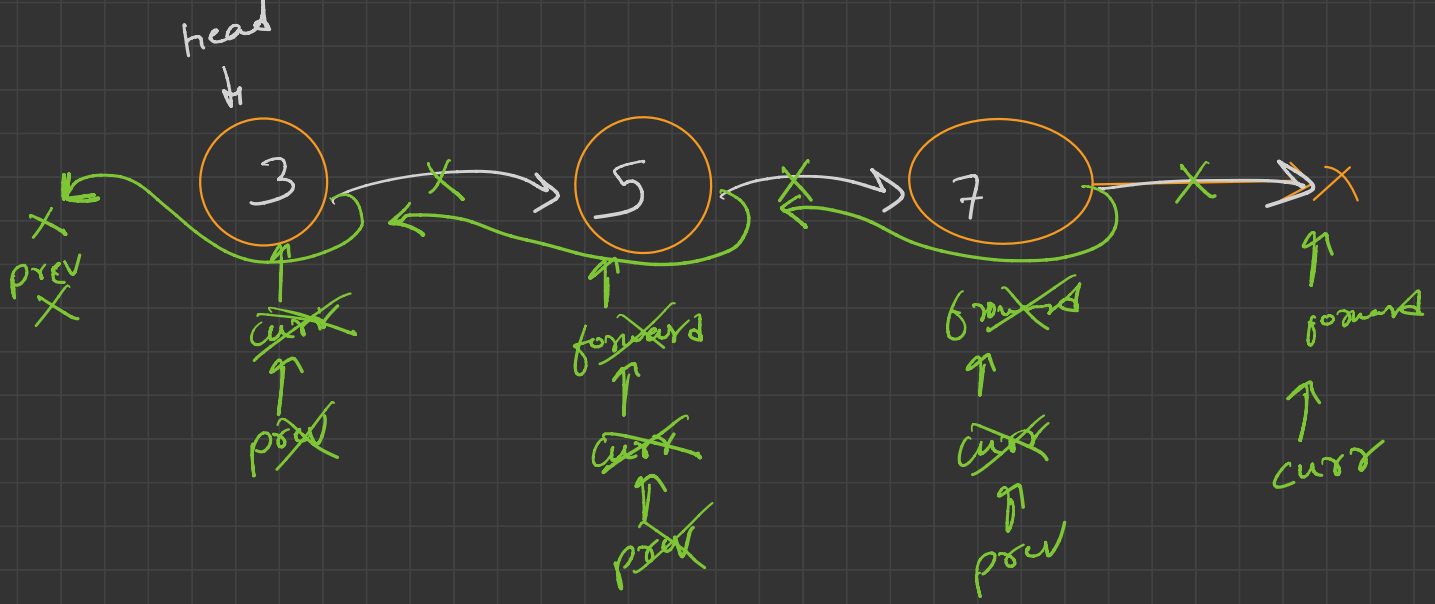
n -> nodes

T.C -> O(n)

S.C -> O(1)

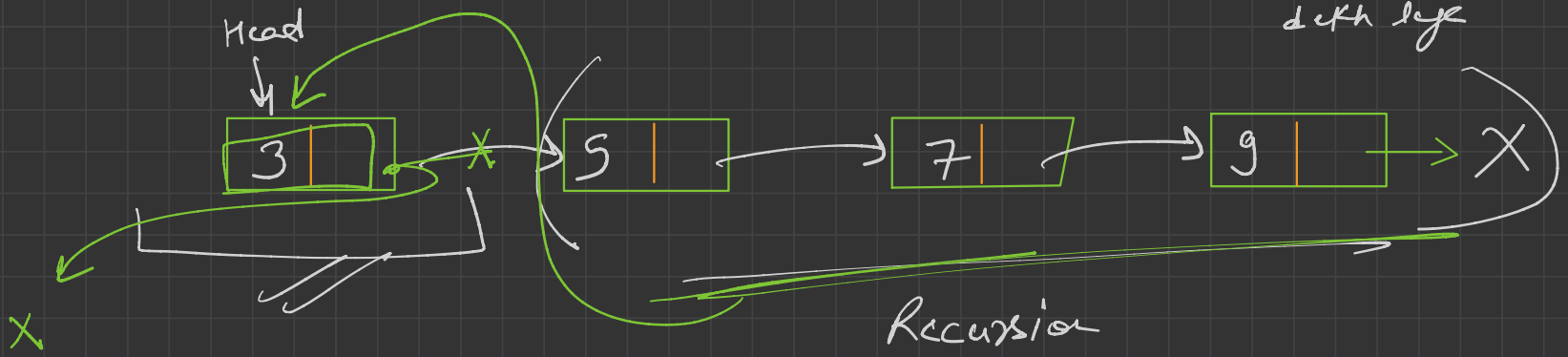
3

return prev;



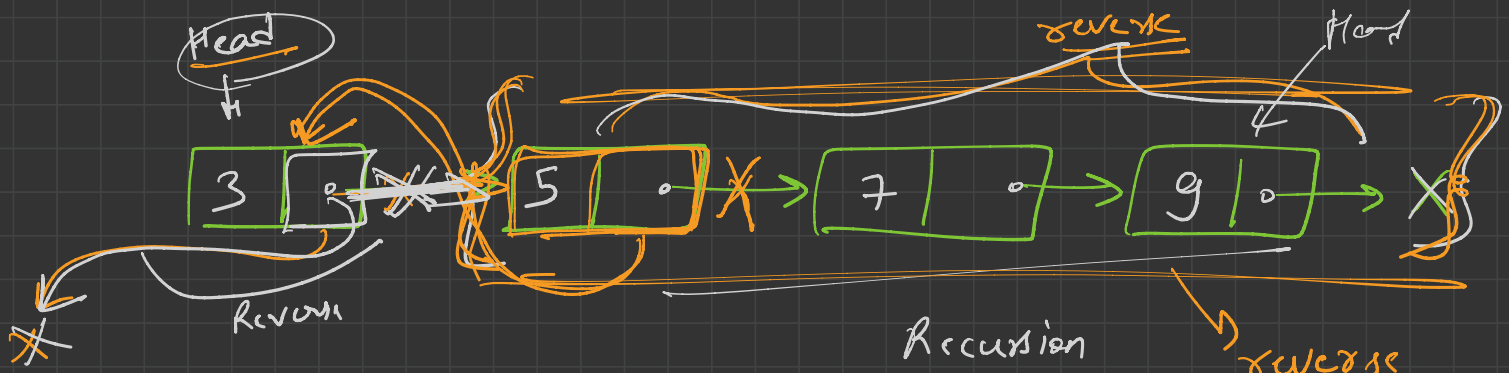
(11) Recursive Sol^n

Recursion → 1 case solve karlo
↓
Baaki recursion dekh lega



→ Recursive Solution :—

I can solve

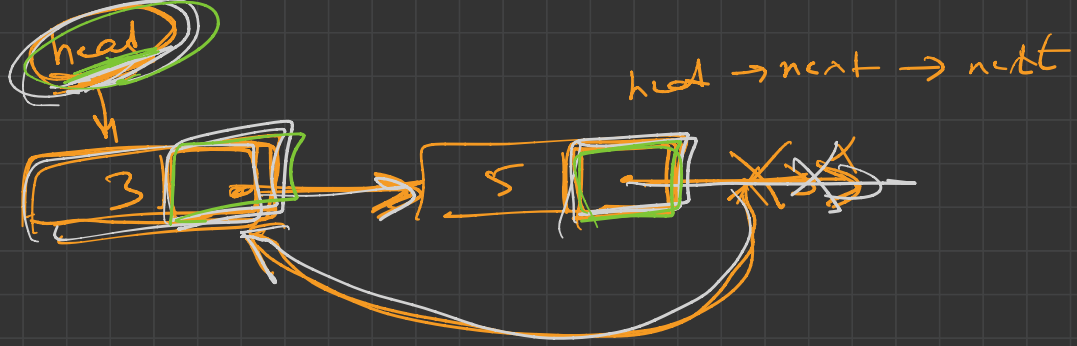


Algo

↳ Base Case :-

↳ 2 kaam

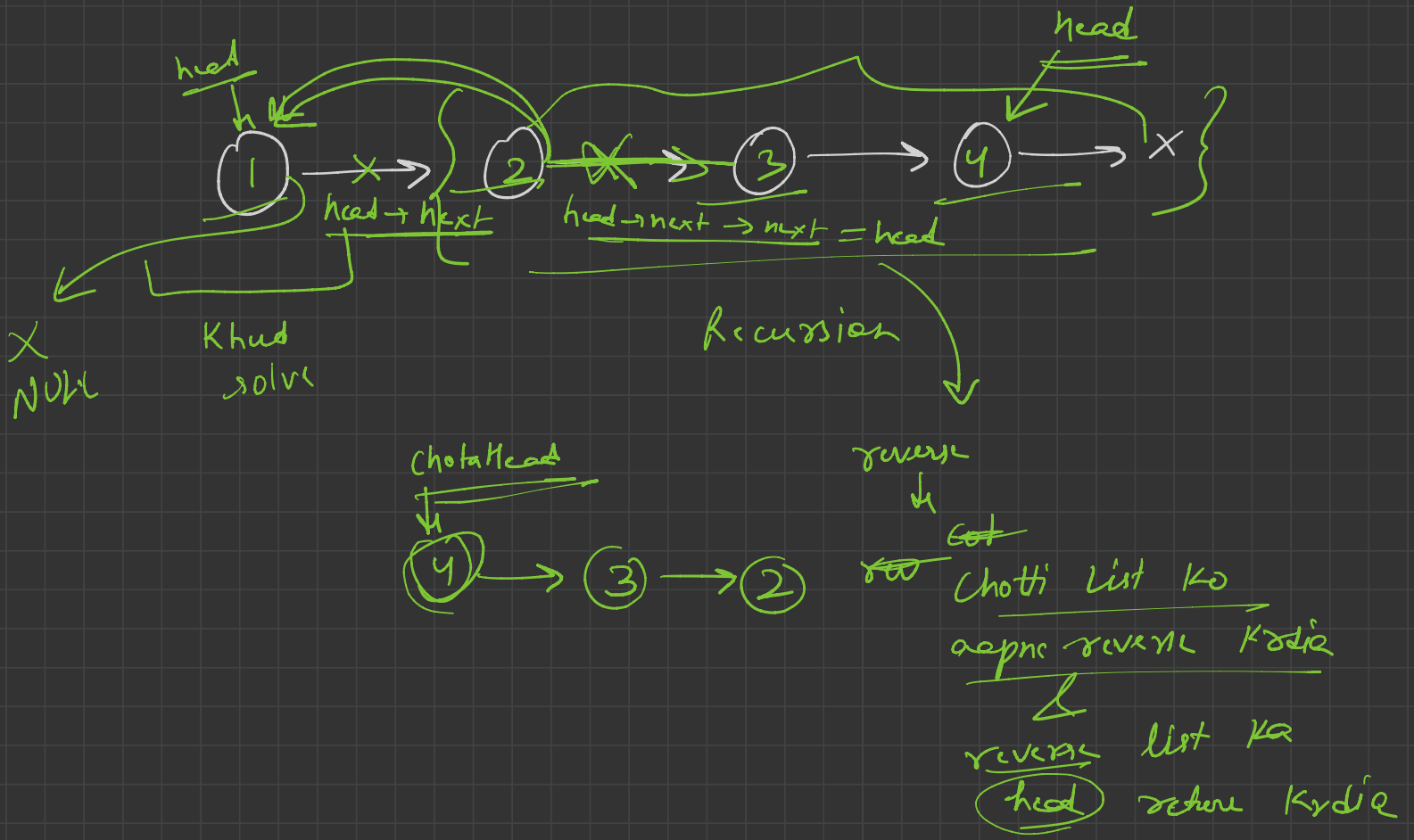
9 → 7 → 5 → 3



head → 3

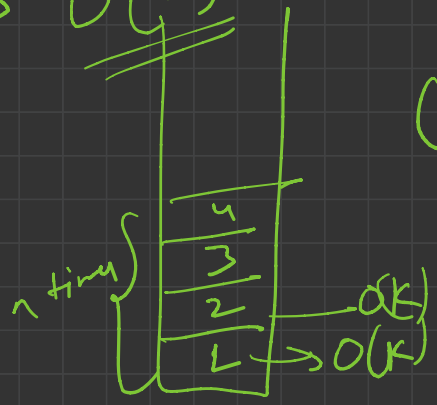
head → next = 5

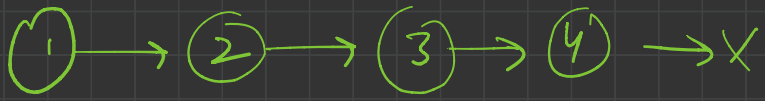
head → next → next = head;


head
1
2
3
4
head
head → next
head → next → next = head
X
NULL
Khud
solve
Recursion
ChotaHead
4
3
2
reverse
Got
Chotti List ko
aapne reverse krdia
reverse list ka
head return krdia

T.C → O(n) → no of nodes in List

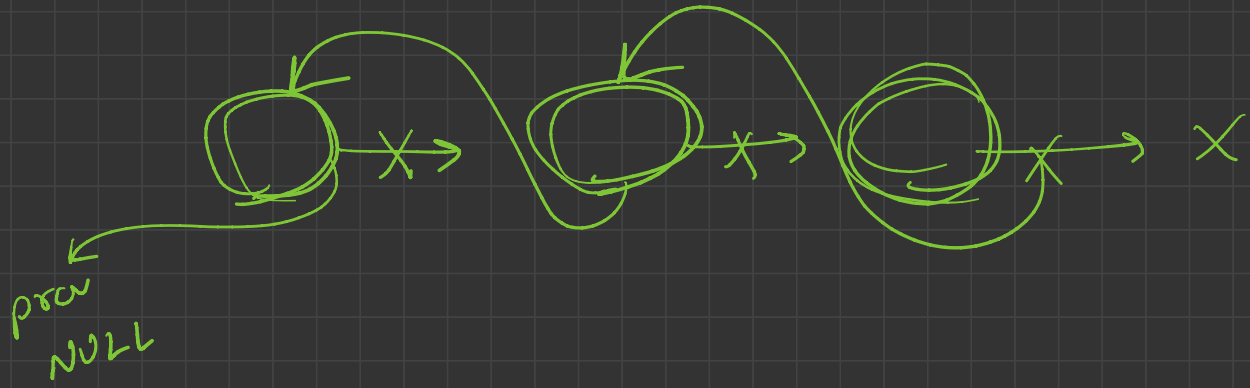
S.C → O(n)





0 1 < n

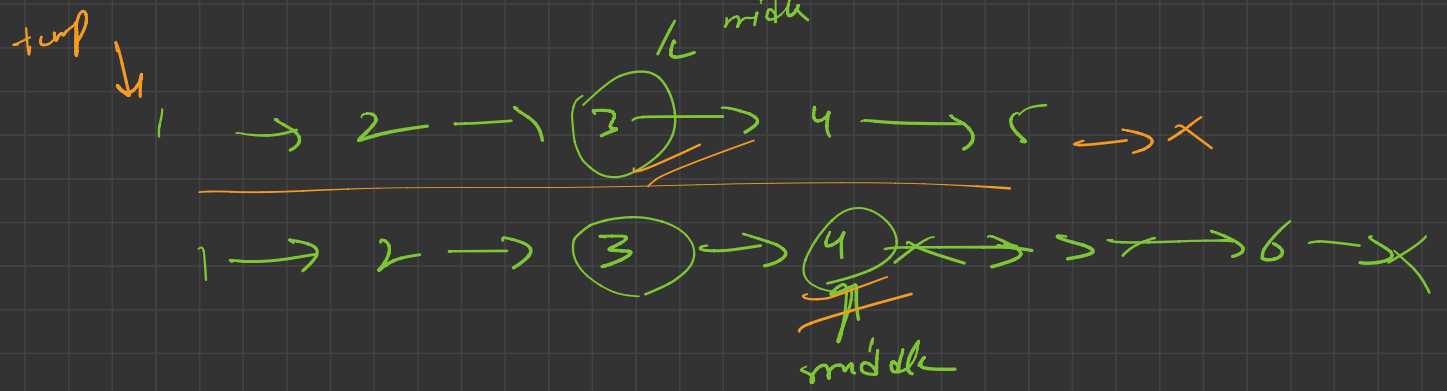
S.C → O(n)



→ H/W → Dobly Linked List

↳ Reverse Karwadena

(11) Middle of Linked List :-



Approach :-

Approach I:

traverse

Length of L'L

Len = even / odd

middle

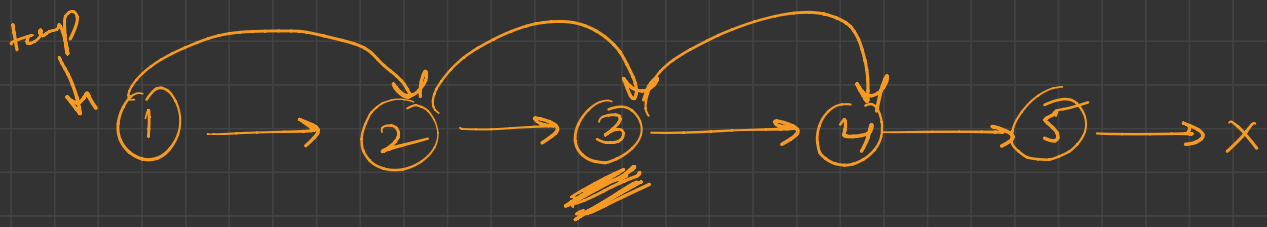
$\frac{len}{2} + 1$

$mid = \frac{len}{2} = \frac{5}{2} = 2 + 1$

$= \frac{6}{2} = 3$

len → odd → $\frac{len}{2} + 1$

→ even = $\frac{len}{2} + 1$



len = 5

ans = $\frac{len}{2}$ + 1 = $\frac{5}{2}$ + 1 = 2 + 1 = 3

temp = head

cnt = 0

while (cnt < ans)

→ getLength() → Entire → T.C → O(n)

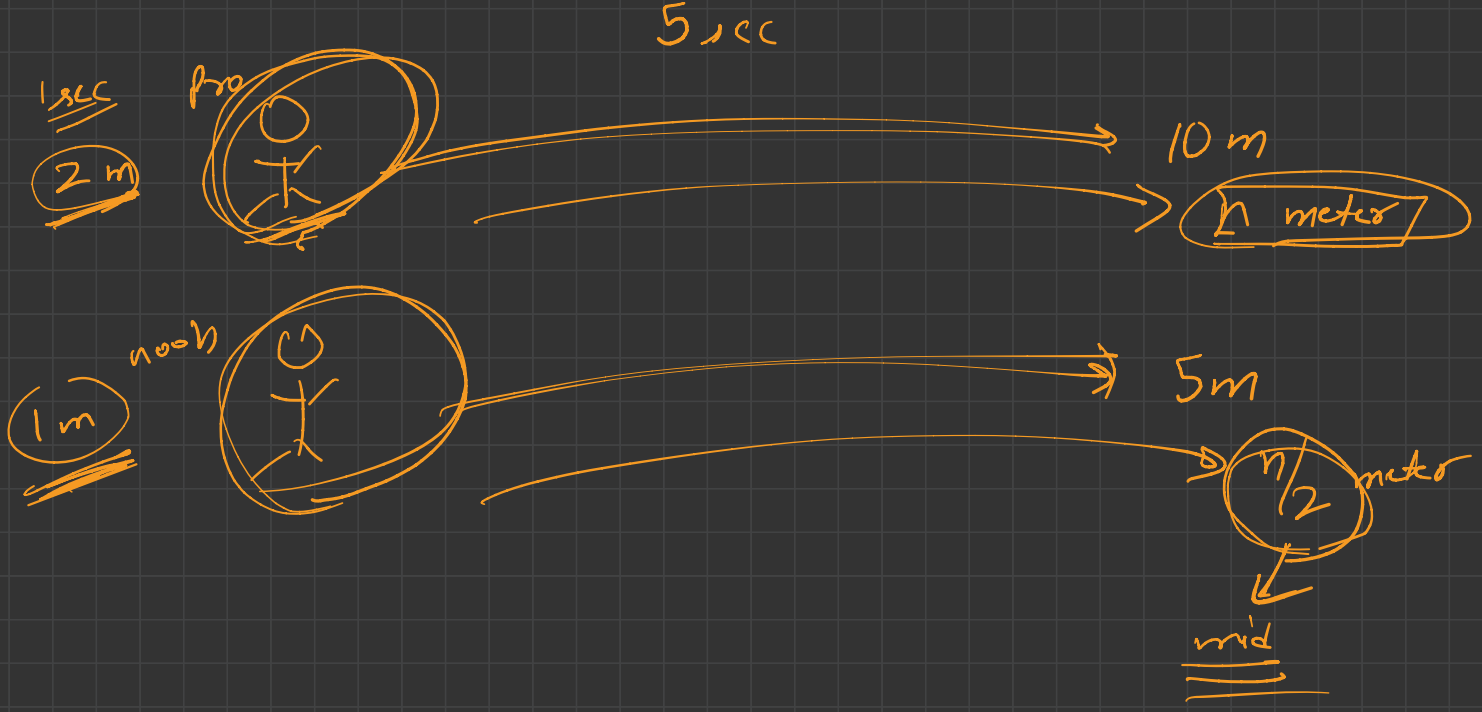
T.C → $O(n) + O(\frac{n}{2})$

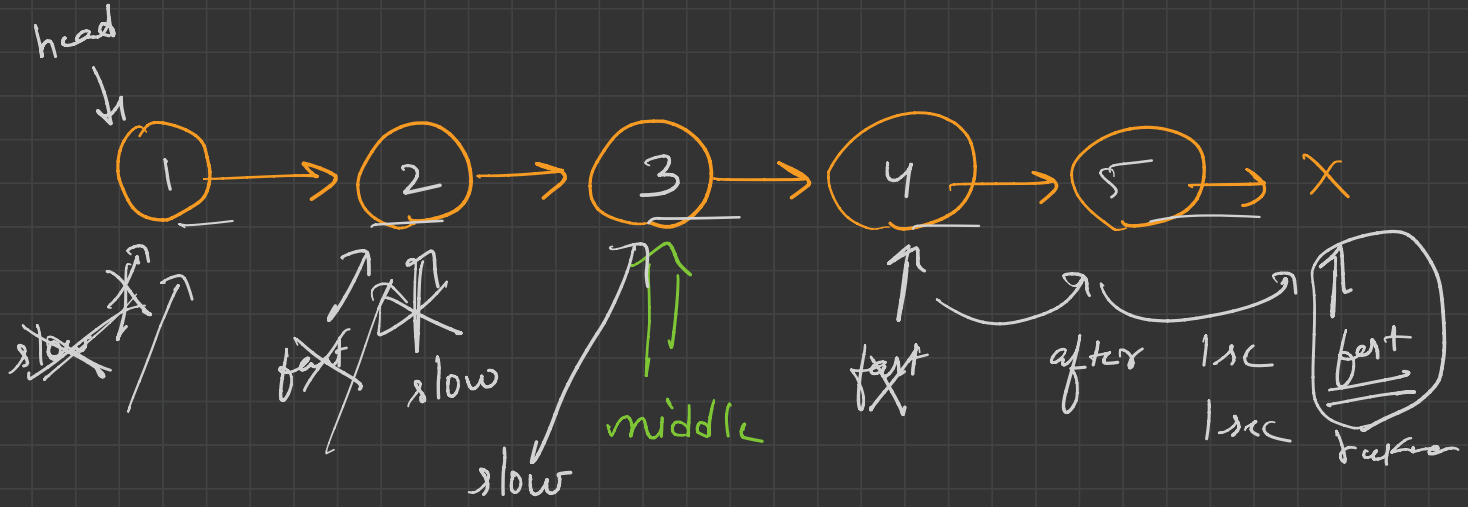
$= O(n) + O(n)$

$= 2O(n)$

T.C → $O(n)$

Optimised Soln





① if → empty list → return NULL

② 1 node → head

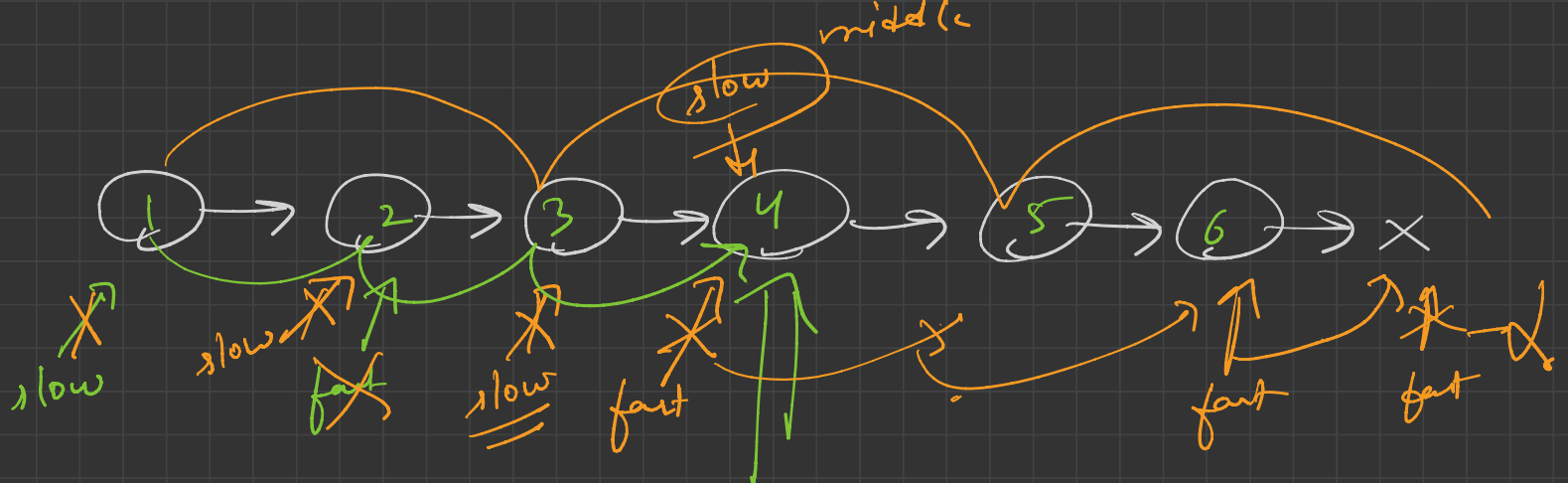
③ 2 node → head → next

④ Algo

Node* fast = head -> next

Node* slow = head

while (fast != NULL)



middle

1 sec

1 sec

T.C → $O(n/2) \rightarrow O(n)$

S.C → $O(1)$

H/w → DLL → reverse