Middle of Linked List

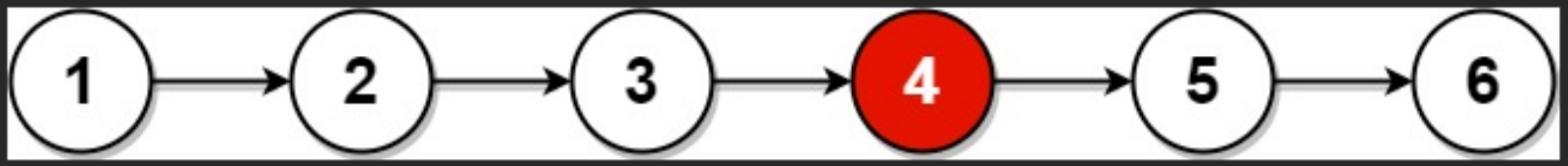
Given the `head` of a singly linked list, return *the middle node of the linked list*.

If there are two middle nodes, return **the second middle** node.

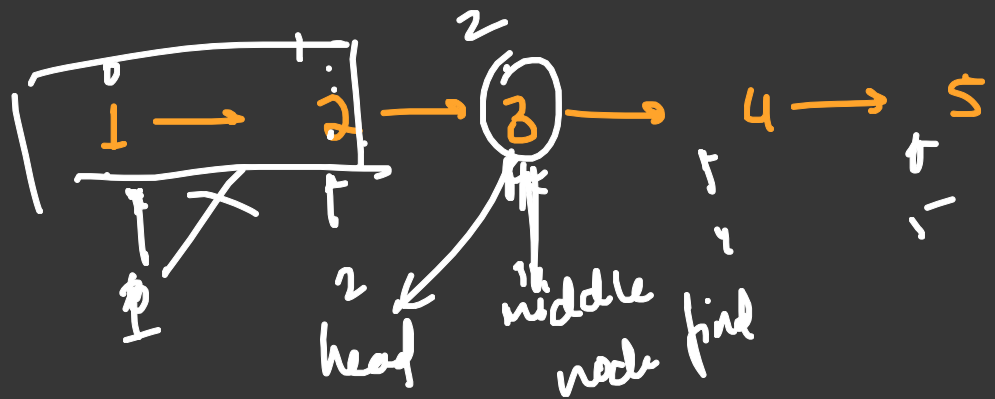
Example 1:

> **Input:** head = [1,2,3,4,5]
> **Output:** [3,4,5]
> **Explanation:** The middle node of the list is node 3.

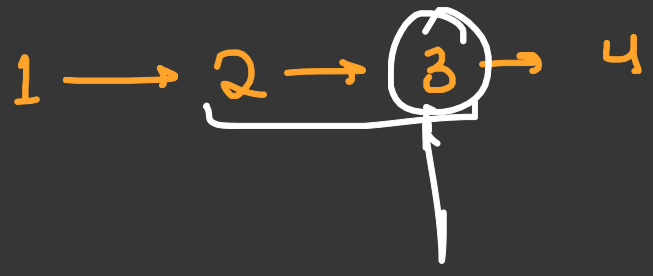
Example 2:



> **Input:** head = [1,2,3,4,5,6]
> **Output:** [4,5,6]
> **Explanation:** Since the list has two middle nodes with values 3 and 4, we return the second one.



3 → 4 → 5



3 → 4

Naive Approach: $n/2$

count = 0

while (temp != NULL) → temp = 5/2 = 2 → O(N)

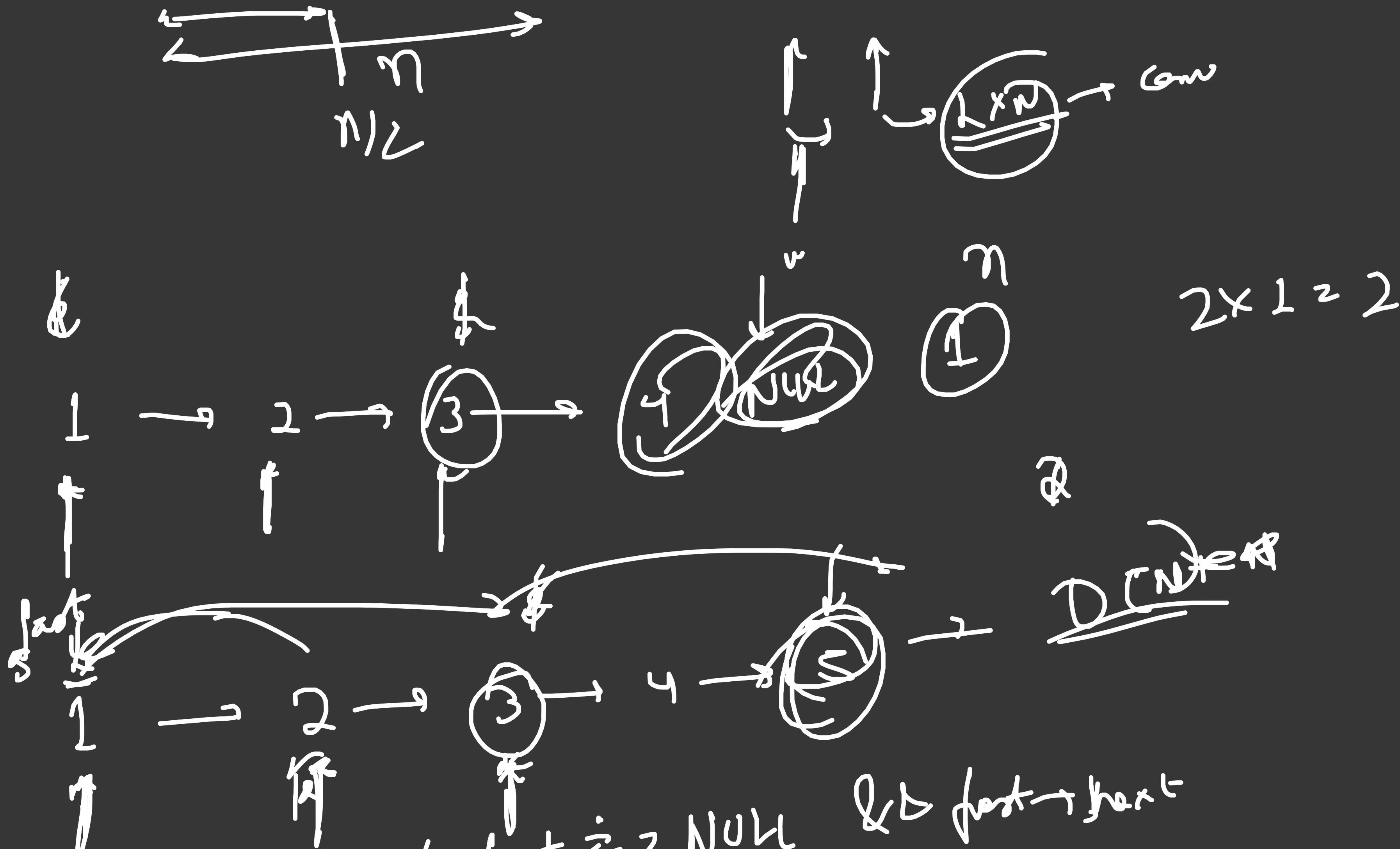
count = 0 → 2

num

O(N + N/2)

O(N/2) = $\frac{2N+0}{2}$

O($\frac{3}{2}$N)




```
while ( fast != NULL && fast->next )
slow = slow->next
fast = fast->next->next
```

Optimal Approach:

- We use Tortoise and Hare algo in which we use two pointer slow and fast.

slow = slow → next → 1skip

fast = fast → next → next → 2skip.

Idea is that when fast complete the list the slow will reach middle as fast is moving with 2X speed of slow speed.

Pseudo code:

```
Node* slow=head, fast=head

while( fast && fast->next)
    slow = slow->head, fast = fast->next->next;

return slow;
```

- Another Way Using Space we can push linked list in array of Node and find return middle node. len/2.