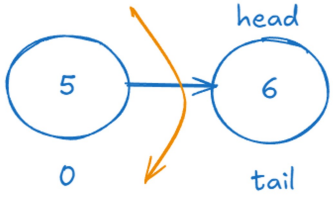
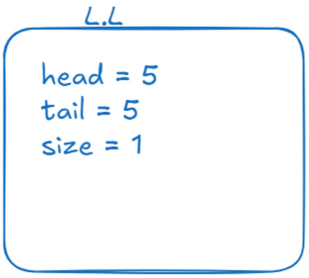
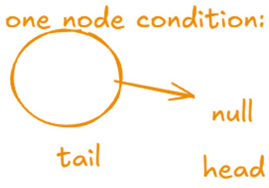
LinkedList [LeetCode] - [Day 14]

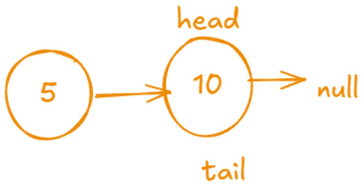
deleteAt(idx)

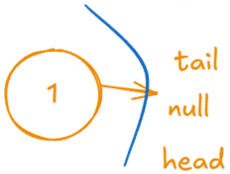


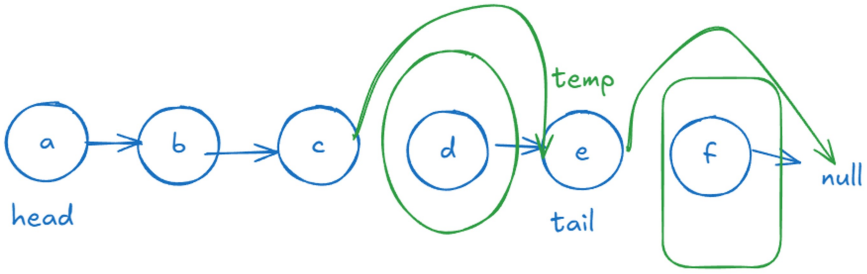





```
if(index == 0){
    head = head.next;
    if(head == null){
        tail = null;
    }
}
```







temp.next = temp.next.next[e]

237. Delete Node in a Linked List

Solved

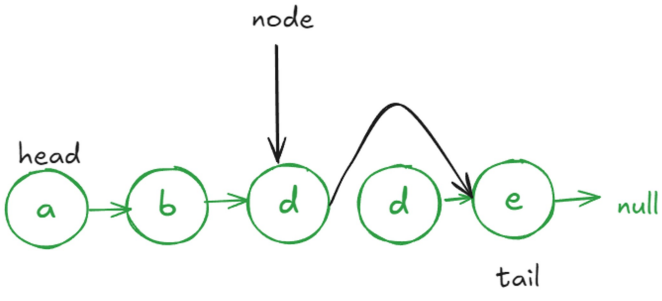
Medium | Topics | Companies

There is a singly-linked list `head` and we want to delete a node `node` in it.

You are given the node to be deleted `node`. You will **not be given access** to the first node of `head`.

All the values of the linked list are **unique**, and it is guaranteed that the given node `node` is not the last node in the linked list.

Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:



node `node` is not the last node in the linked list.

Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

- The value of the given node should not exist in the linked list.
- The number of nodes in the linked list should decrease by one.
- All the values before `node` should be in the same order.
- All the values after `node` should be in the same order.

node.data = node.next.data

node.next = node.next.next

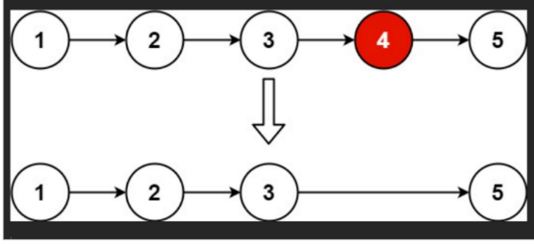
19. Remove Nth Node From End of List

Solved

Medium | Topics | Companies | Hint

Given the `head` of a linked list, remove the `n`th node from the end of the list and return its head.

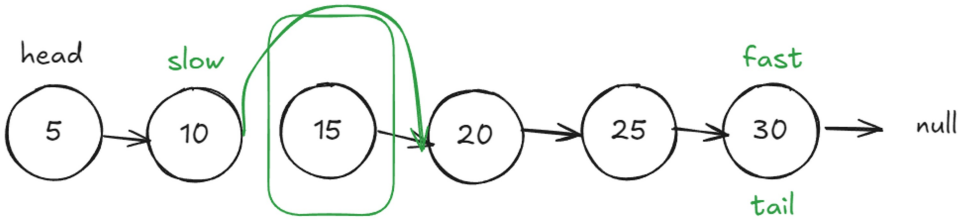
Example 1:



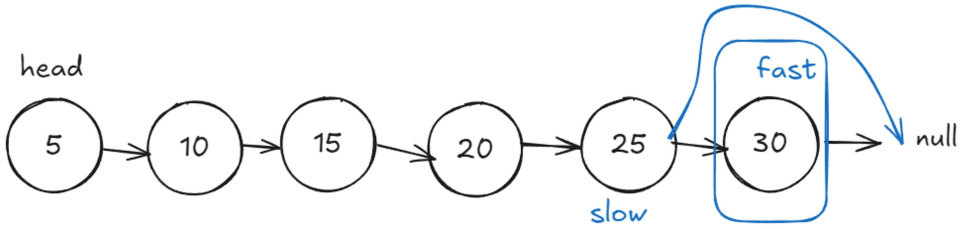

```java
public ListNode removeNthFromEnd(ListNode head, int n) {

}
```

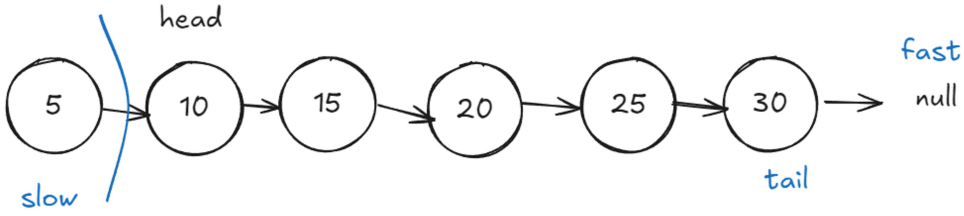
Nth Node from the last = 4



fast.next == null
[stopping]

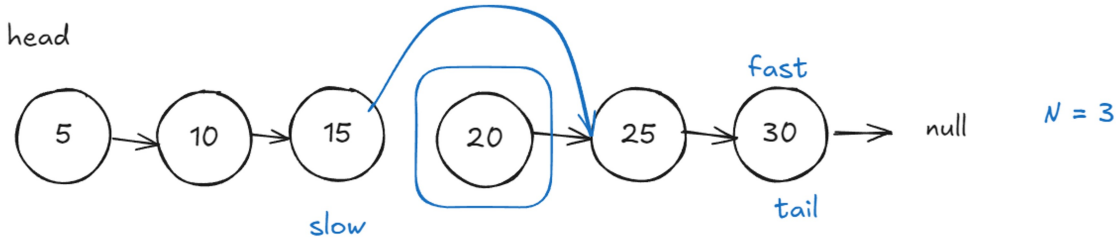


N = 1



N = 6

return head = head.next;



return head = head.next;

```java
public ListNode removeNthFromEnd(ListNode head, int n) {
    ListNode slow = head;
    ListNode fast = head;

    for(int i = 0 ; i < n ; i++){
        fast = fast.next;
    }

    if(fast == null){
        return head.next;
    }
    while(fast.next != null){
        slow = slow.next;
        fast = fast.next;
    }
    slow.next = slow.next.next;

    return head;
}
```

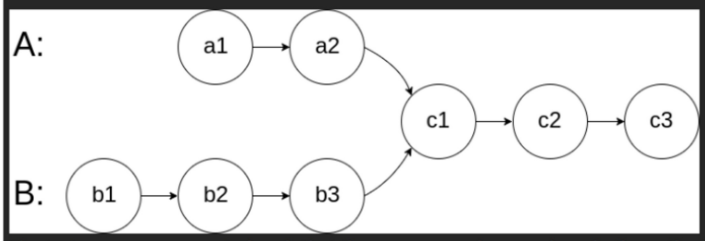
160. Intersection of Two Linked Lists

Solved

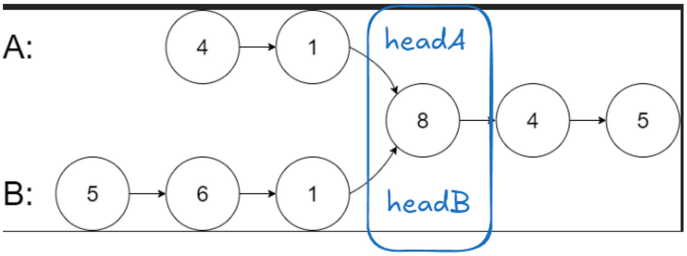
Easy | Topics | Companies

Given the heads of two singly linked-lists `headA` and `headB`, return *the node at which the two lists intersect*. If the two linked lists have no intersection at all, return `null`.

For example, the following two linked lists begin to intersect at node `c1`:

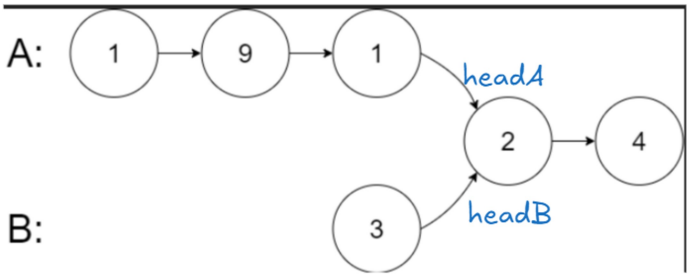


The test cases are generated such that there are no cycles anywhere in the entire linked structure.



length() of lenA - 5
lenB = 6 -- 5

lenA != lenB



lenA = 5 -- 3
lenB = 3

Create a Length Function : HeadA , HeadB -> length of individual L.L.

```
while(lenA > lenB){
    headA = headA.next
    lenA--;
}
while(lenB > lenA){
    headB = headB.next
    lenB--;
}
while(headA != headB){
    headA = headA.next;
    headB = headB.next;
}
return headA or headB;
```

```java
    private int getLength(ListNode node){
        int length = 0;
        while(node != null){
            length++;
            node = node.next;
        }
        return length;
    }
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
        int lenA = getLength(headA);
        int lenB = getLength(headB);

        while(lenA > lenB){
            headA = headA.next;
            lenA--;
        }
        while(lenB > lenA){
            headB = headB.next;
            lenB--;
        }
        while(headA != headB){
            headA = headA.next;
            headB = headB.next;
        }
        return headA;
    }
```

876. Middle of the Linked List

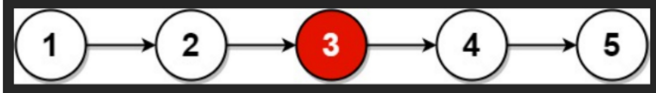
Solved

Easy | Topics | Companies

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

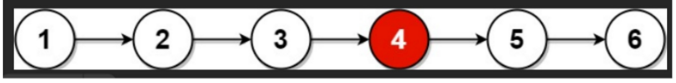
If there are two middle nodes, return **the second middle** node.

Example 1:

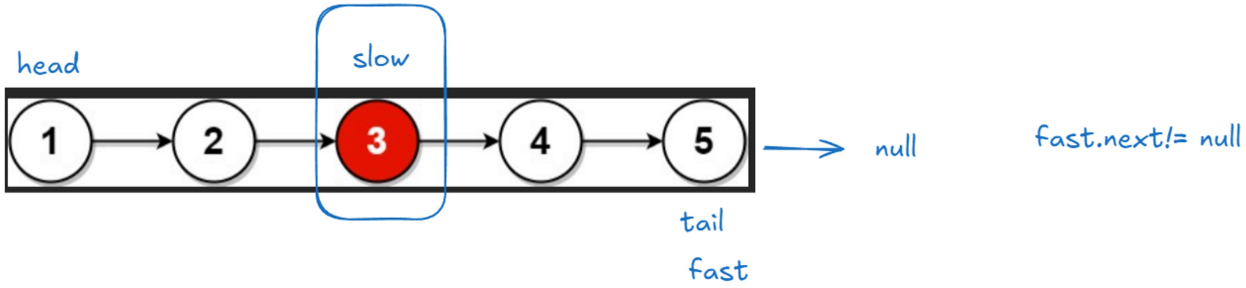


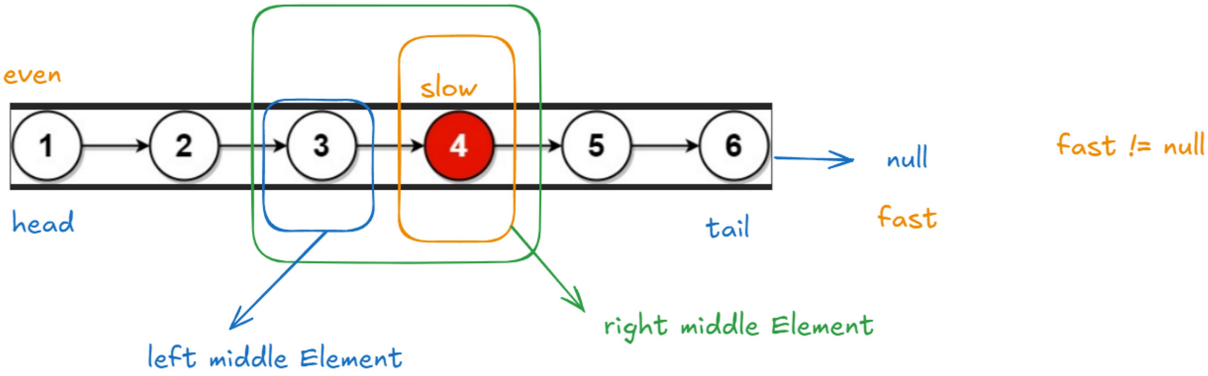
Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.

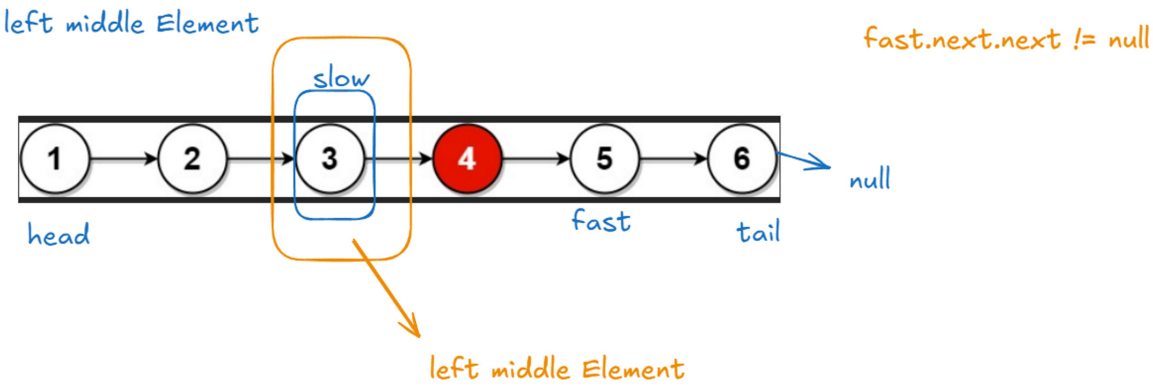
Example 2:



odd :


head
slow
1
2
3
4
5
null
fast.next!= null
tail
fast


even
slow
1
2
3
4
5
6
null
fast != null
head
tail
fast
left middle Element
right middle Element


left middle Element
fast.next.next != null
slow
1
2
3
4
5
6
null
head
fast
tail
left middle Element

```java
class Solution {
    public ListNode middleNode(ListNode head) {
        ListNode slow = head;
        ListNode fast = head;
        while(fast != null && fast.next != null){
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }
}
```

Change Status

Completed

Attendance Code: 23E3FEDB

Close Update

2095. Delete the Middle Node of a Linked List

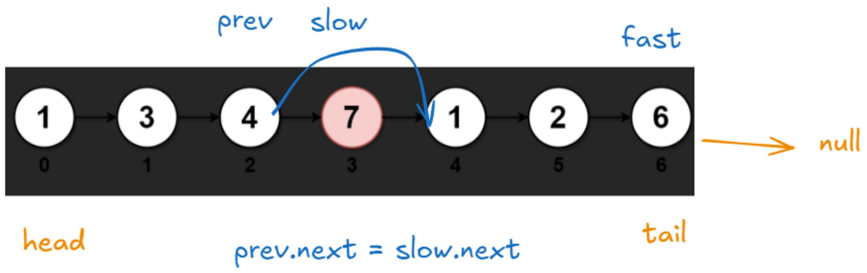
Solved

Medium Topics Companies Hint

You are given the `head` of a linked list. **Delete** the **middle node**, and return *the* `head` *of the modified linked list.*

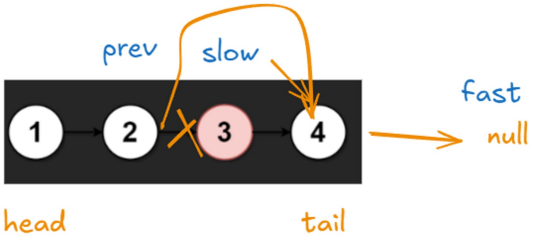
The **middle node** of a linked list of size `n` is the `⌊n / 2⌋th` node from the **start** using **0-based indexing**, where `⌊x⌋` denotes the largest integer less than or equal to `x`.

- For `n` = `1`, `2`, `3`, `4`, and `5`, the middle nodes are `0`, `1`, `1`, `2`, and `2`, respectively.




```java
ListNode slow = head;
ListNode fast = head;
ListNode prev = null;

while(fast != null && fast.next != null){
    prev = slow;
    slow = slow.next;
    fast = fast.next.next;
}
```



prev.next = slow.next


```java
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if(head == null || head.next == null){
            return null;
        }
        ListNode slow = head;
        ListNode fast = head;
        ListNode prev = null;

        while(fast != null && fast.next != null){
            prev = slow;
            slow = slow.next;
            fast = fast.next.next;
        }

        prev.next = slow.next;
        return head;
    }
}
```