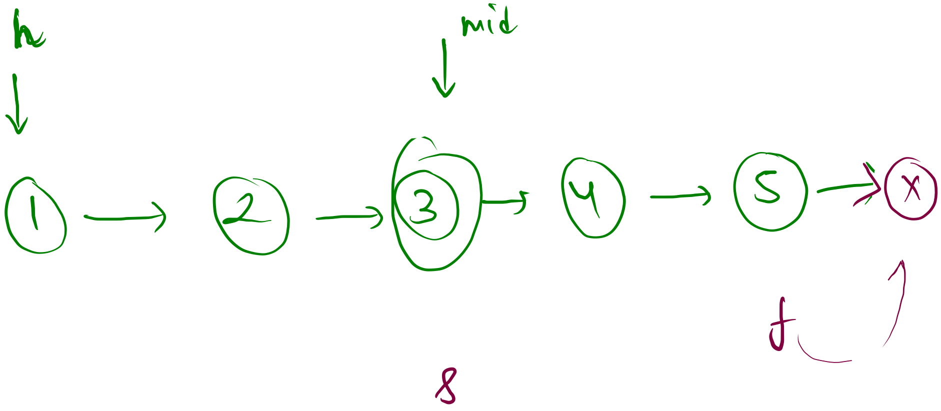
linked list. (mid).



even

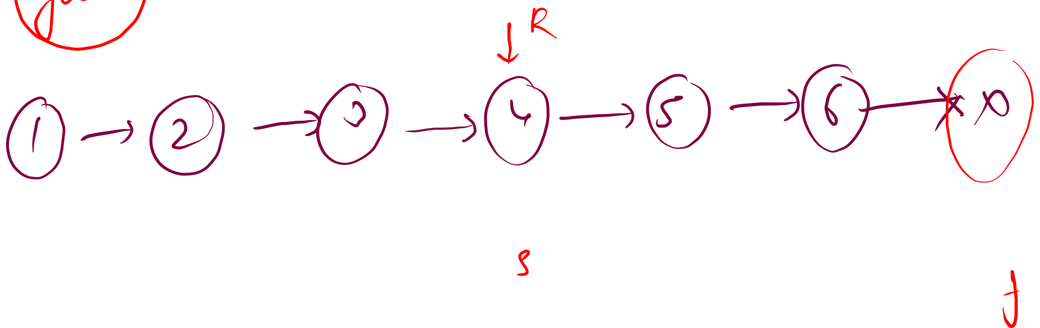
* f != null

false

odd

4d

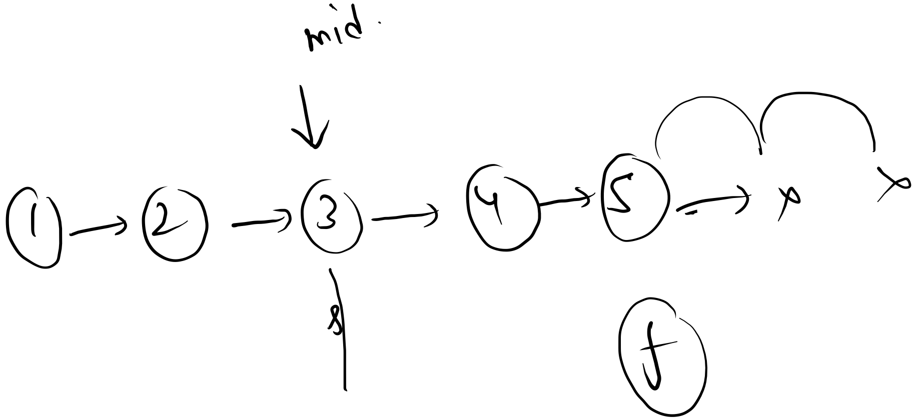
f.next != null



2

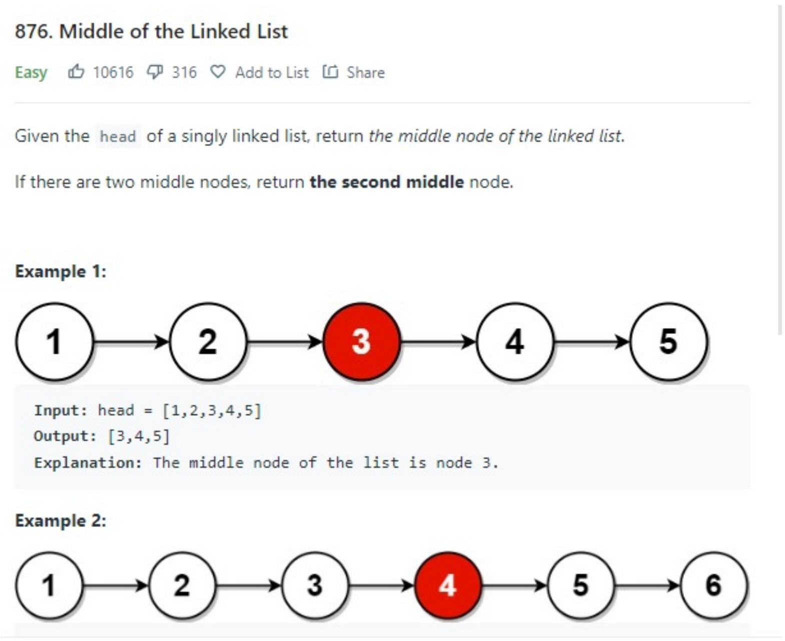
left wala. mid.

even



1
fast.next != null

2
fast.next.next != null



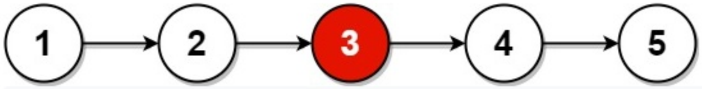
876. Middle of the Linked List

Easy 10616 316 Add to List Share

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

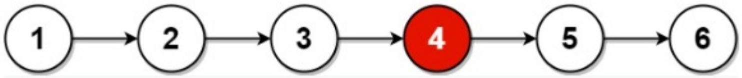
If there are two middle nodes, return **the second middle** node.

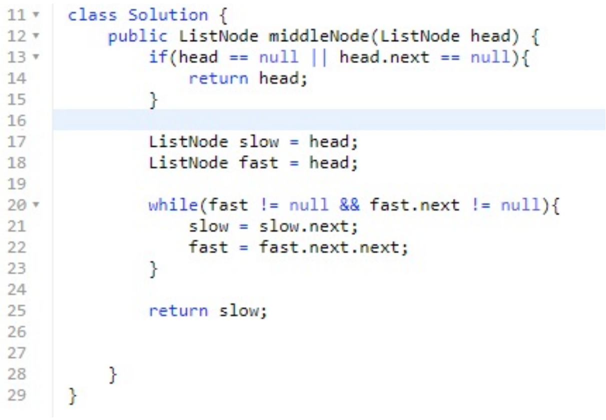
Example 1:



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

Example 2:



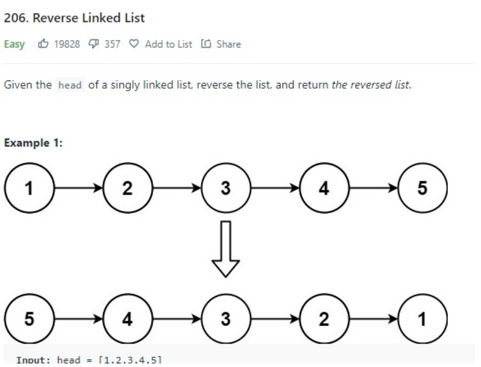


```
class Solution {
    public ListNode middleNode(ListNode head) {
        if(head == null || head.next == null){
            return head;
        }

        ListNode slow = head;
        ListNode fast = head;

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

        return slow;


    }
}
```

LC- 876 $\rightarrow$ mid (R)

LC-206 $\rightarrow$ reverse

LC-21 $\rightarrow$ merge 2 sorted



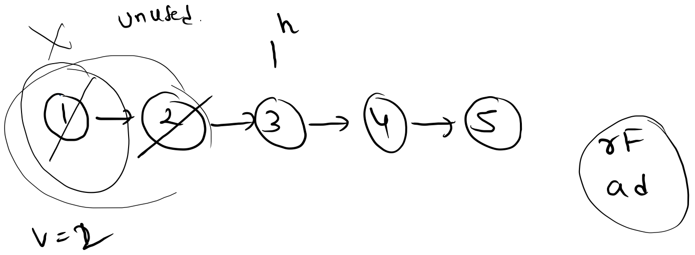
206. Reverse Linked List

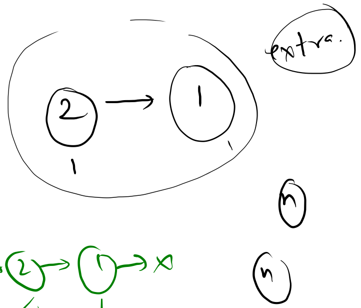
Easy 19828 357 Add to List Share

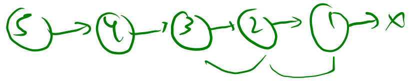
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

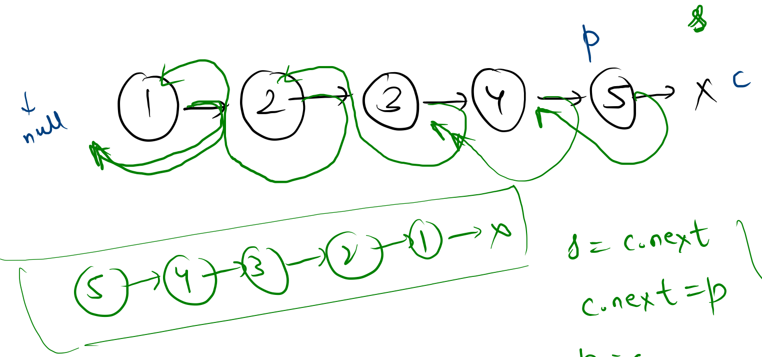
Example 1:

```
Input: head = [1,2,3,4,5]
```









while
c != null

process

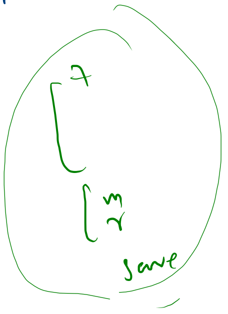
s = c.next

c.next = p

p = c

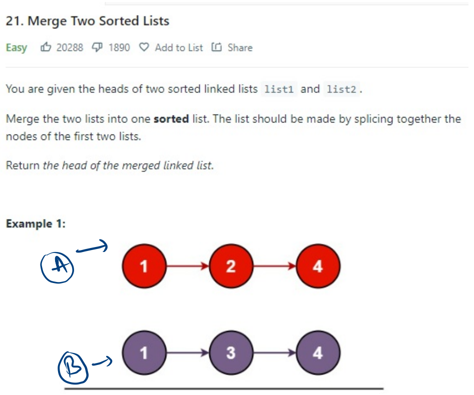
c = s

process



h = p

return p


21. Merge Two Sorted Lists

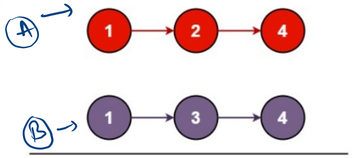
Easy 20288 1890 Add to List Share

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:



Case 1:

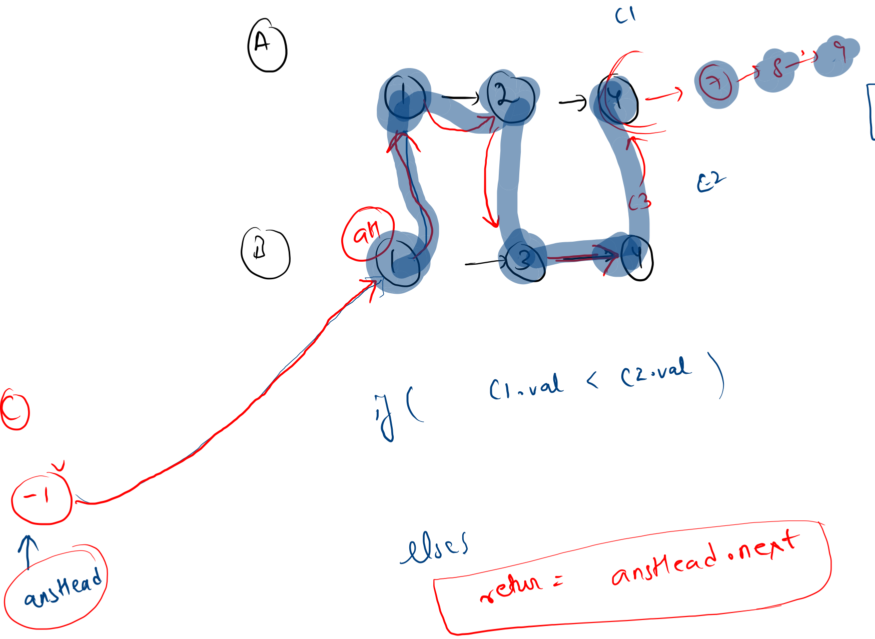
A ↳ size==0 B ↳ size not 0

↳ B.head

Case 2:

A s≠0 B s==0

↳ A.head



A exsists

[

B exsists.

c1 != null &&

c2 != null

if (c1.val < c2.val)

elses

return = ansHead.next

}