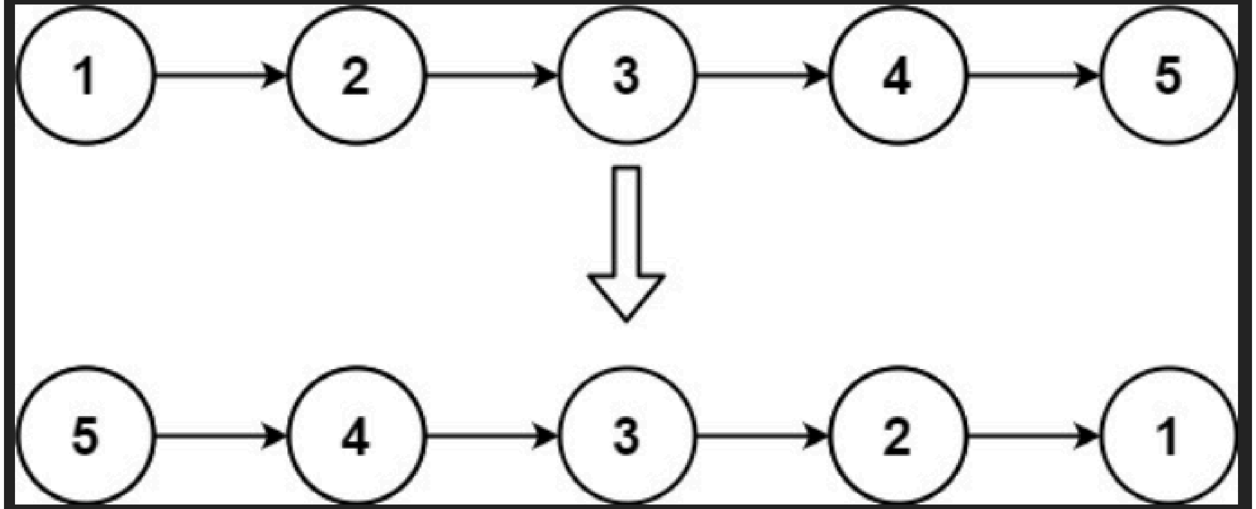
206. Reverse Linked List

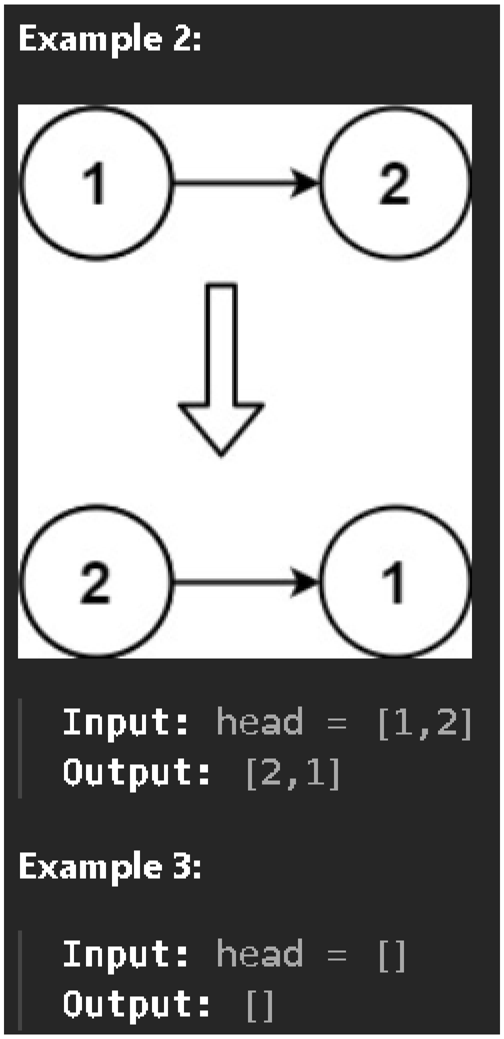
Easy · Topics · Companies

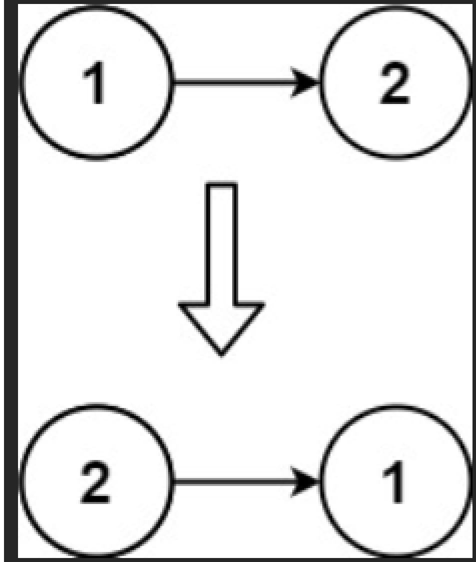
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:



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

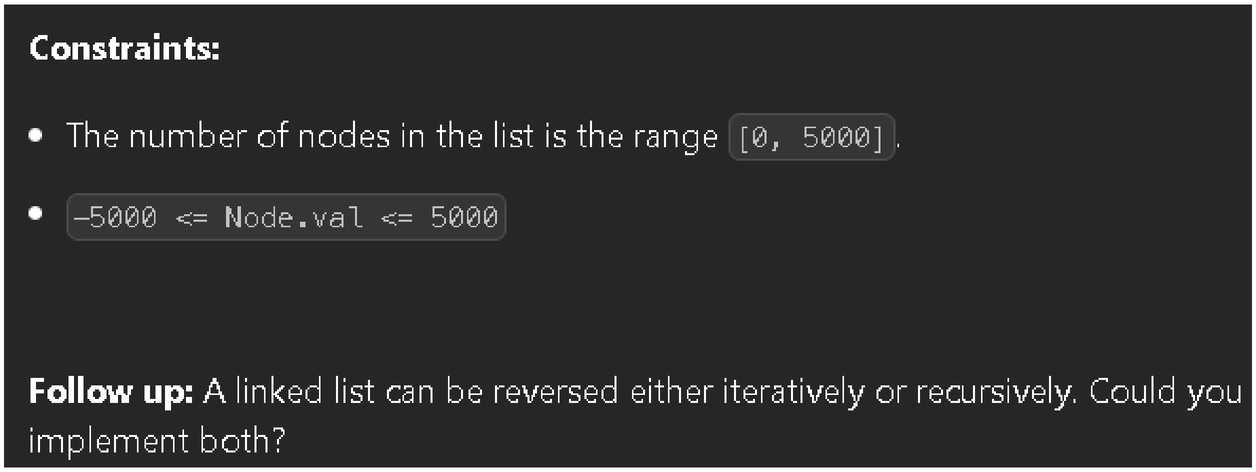

Example 2:



```
Input: head = [1,2]
Output: [2,1]
```

Example 3:

```
Input: head = []
Output: []
```


Constraints:

- The number of nodes in the list is the range `[0, 5000]`.
- `-5000 <= Node.val <= 5000`

Follow up: A linked list can be reversed either iteratively or recursively. Could you implement both?

Python:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
```

```
#        self.val = val
#        self.next = next

from typing import Optional

class Solution:
    def reverseList(self, head: Optional['ListNode']) -> Optional['ListNode']:
        # Iterative approach
        prev = None
        current = head
        while current:
            nxt = current.next      # Store next node
            current.next = prev     # Reverse the link
            prev = current          # Move prev forward
            current = nxt           # Move current forward
        return prev   # New head of the reversed list


    def reverseListRecursive(self, head: Optional['ListNode']) -> Optional['ListNode']:
        # Recursive approach
        if not head or not head.next:
            return head
        new_head = self.reverseListRecursive(head.next)
        head.next.next = head
        head.next = None
        return new_head
```

JavaScript:

```
/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @return {ListNode}
 */
var reverseList = function(head) {
    let prev = null;
    let curr = head;
    while (curr !== null) {
        let nextTemp = curr.next; // store next node
        curr.next = prev;         // reverse link
        prev = curr;              // move prev forward
        curr = nextTemp;          // move curr forward
    }

    return prev; // new head
};
```

Java:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    // Iterative Solution
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;      // Previous node starts as null
        ListNode curr = head;      // Current node starts as head

        while (curr != null) {
            ListNode nextNode = curr.next; // store next node
            curr.next = prev;              // reverse pointer
            prev = curr;                   // move prev forward
            curr = nextNode;               // move curr forward
        }
        return prev;   // new head
    }

    // Recursive Solution (Follow-up)
    public ListNode reverseListRecursive(ListNode head) {
        // Base case: empty list or single node
        if (head == null || head.next == null) {
            return head;
        }
        // Reverse the rest of the list
        ListNode newHead = reverseListRecursive(head.next);

        // Reverse the current node's pointer
        head.next.next = head;
        head.next = null;

        return newHead;
    }
}
```