

■ Reverse a Linked List

The most important linked list operation — iterative + recursive

● Linked List

■ EASY

■ What You Will Learn

- The three-pointer iterative reversal technique
- Recursive reversal and how the call stack helps
- Why you need prev, curr, and next pointers
- Dry run — watching the pointers dance

■ Intuition First

Reversing a linked list is like reversing a chain of people holding hands. You go to each person and make them grab the hand of the person behind them instead. Three people involved at each step: the one behind (prev), the current person, and the one ahead (next). Save 'next' before you redirect, or you'll lose the chain!

```
Original: null ← prev  curr → next
           1 → 2 → 3 → 4 → null

Step 1: prev=null, curr=1, next=2
      curr.next = prev → 1 → null
      prev=1, curr=2

Step 2: prev=1, curr=2, next=3
      curr.next = prev → 2 → 1 → null
      prev=2, curr=3

Step 3: prev=2, curr=3, next=4
      curr.next = prev → 3 → 2 → 1 → null
      prev=3, curr=4

Step 4: prev=3, curr=4, next=null
      curr.next = prev → 4 → 3 → 2 → 1 → null
      prev=4, curr=null

curr==null → return prev=4 as new head ✓
```

■ Java Code — Iterative

```
1  class Solution {
2  public ListNode reverseList(ListNode head) {
3      ListNode prev = null;
4      ListNode curr = head;
5  }
```

```

6   while (curr != null) {
7       ListNode next = curr.next; // Save next FIRST!
8       curr.next = prev; // Reverse the pointer
9       prev = curr; // Move prev forward
10      curr = next; // Move curr forward
11  }
12  return prev; // prev is now the new head
13  }
14  }

```

■ Recursive Version

```

1   public ListNode reverseList(ListNode head) {
2       if (head == null || head.next == null) return head;
3
4       ListNode newHead = reverseList(head.next); // Recurse
5       head.next.next = head; // Make next point back to head
6       head.next = null; // Cut forward link
7       return newHead;
8   }

```

■■ Complexity

Type	Value	Meaning
Time	O(n)	Visit each node once
Space	O(1) iterative / O(n) recursive	Iterative uses 3 vars; recursive uses call stack

■■ Most Common Mistake

Forgetting to save curr.next BEFORE doing curr.next=prev. If you redirect first, you lose the rest of the list forever!

■ Quick Revision

✓	Three pointers: prev=null, curr=head, next
✓	Save next BEFORE redirecting: next=curr.next
✓	Redirect: curr.next=prev
✓	Advance: prev=curr, curr=next
✓	Return prev (new head) when curr==null