21. Merge Two Sorted Lists

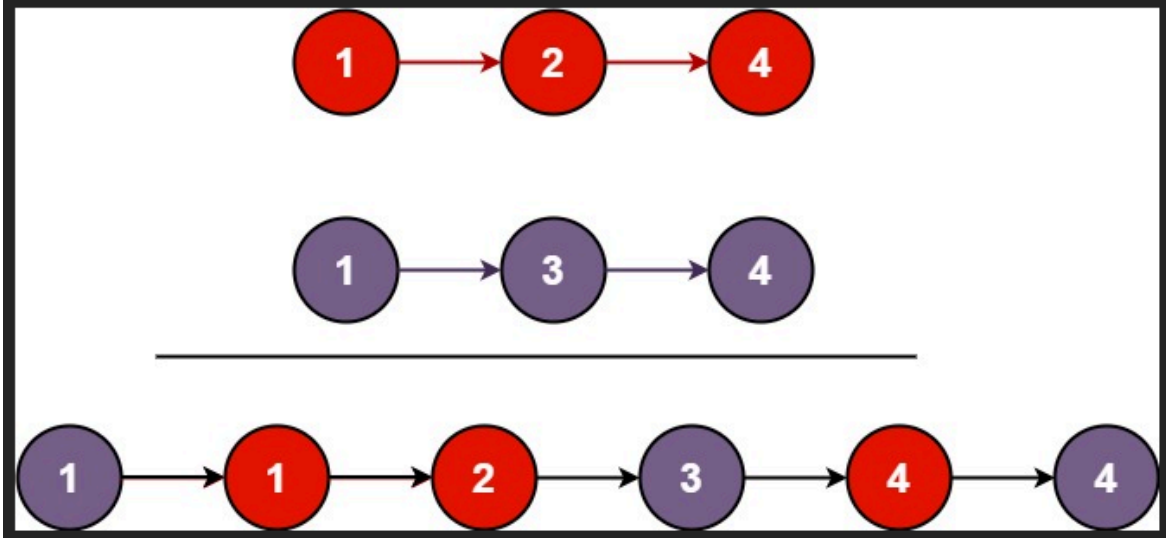
Easy | Topics | Companies

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:




```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]
Output: [0]
```

Constraints:

- The number of nodes in both lists is in the range `[0, 50]`.
- `-100 <= Node.val <= 100`
- Both `list1` and `list2` are sorted in **non-decreasing** order.

Python3:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) ->
Optional[ListNode]:
        # Dummy node to simplify result construction
        dummy = ListNode(-1)
        current = dummy

        # Traverse both lists
        while list1 and list2:
            if list1.val <= list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
```

```
            list2 = list2.next
        current = current.next

    # Attach remaining part (if any)
    current.next = list1 if list1 else list2

    return dummy.next
```

JavaScript:

```
// Definition for singly-linked list.
function ListNode(val, next) {
    this.val = (val===undefined ? 0 : val)
    this.next = (next===undefined ? null : next)
}

/**
 * @param {ListNode} list1
 * @param {ListNode} list2
 * @return {ListNode}
 */
var mergeTwoLists = function(list1, list2) {
    // Create a dummy node to simplify handling the head
    let dummy = new ListNode(-1);
    let current = dummy;

    // Traverse both lists
    while (list1 !== null && list2 !== null) {
        if (list1.val <= list2.val) {
            current.next = list1;
            list1 = list1.next;
        } else {
            current.next = list2;
            list2 = list2.next;
        }
        current = current.next;
    }

    // Attach the remaining part (if any)
    if (list1 !== null) {
        current.next = list1;
    } else {
        current.next = list2;
    }
```

```
    // Return the merged list starting at dummy.next
    return dummy.next;
};

// --------- Example usage ---------

// Helper function to convert array to linked list
function arrayToList(arr) {
    let dummy = new ListNode(-1);
    let current = dummy;
    for (let num of arr) {
        current.next = new ListNode(num);
        current = current.next;
    }
    return dummy.next;
}

// Helper function to convert linked list to array (for easy output)
function listToArray(head) {
    let result = [];
    while (head !== null) {
        result.push(head.val);
        head = head.next;
    }
    return result;
}

// Test cases
let list1 = arrayToList([1,2,4]);
let list2 = arrayToList([1,3,4]);
let merged = mergeTwoLists(list1, list2);
console.log(listToArray(merged)); // [1,1,2,3,4,4]

console.log(listToArray(mergeTwoLists(arrayToList([]), arrayToList([])))); // []
console.log(listToArray(mergeTwoLists(arrayToList([]), arrayToList([0])))); // [0]
```

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 {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        // Dummy node to simplify the process
        ListNode dummy = new ListNode(-1);
        ListNode current = dummy;

        // Traverse both lists
        while (list1 != null && list2 != null) {
            if (list1.val <= list2.val) {
                current.next = list1;
                list1 = list1.next;
            } else {
                current.next = list2;
                list2 = list2.next;
            }
            current = current.next;
        }

        // Attach the remaining nodes (if any)
        if (list1 != null) {
            current.next = list1;
        } else {
            current.next = list2;
        }

        // The merged list starts from dummy.next
        return dummy.next;
    }
}
```