# Merge Two Sorted Lists

**Problem Statement**

Given two sorted linked lists, merge them so that the resulting linked list is also sorted.

Consider two sorted linked lists as an example.

A screenshot of a computer

Description automatically generated with low confidence

The merged linked list should look like this:



**Approach**

We keep a pointer to keep track of the sorted list.

We have two options to start the list:

1. Make the head of the sorted list to whichever value is less in list1 and list2
2. Make a dummy node and run the algorithm normally

These are necessary because we need a starting head for the sorted list to match the algorithm.

The algorithm itself is straightforward, similar to a two-pointer algorithm where we compare two values to see which is lesser.

Whichever current node between the two lists has a lesser value, add it onto the sorted list.

// dummy node

ListNode sorted = new ListNode();

// merge two sorted lists into single sorted list

while(list1 != null && list2 != null)

{

if(list1.val <= list2.val)

{

currSorted.next = list1;

list1= list1.next;

}

else

{

currSorted.next = curr2;

curr2 = curr2.next;

}

currSorted = currSorted.next;

}

We need to check for the edge case that the lists are not the same length.

The loop condition will terminate when one of the lists reaches null (end of the list).

Therefore, we set the end of the sorted list to the head of the list that still had length, because the ordering of values will still hold to be true.

currSorted.next = (curr1 == null ? curr2 : curr1);

Remember to return the actual head and not the dummy node

return sorted.next;

**Runtime Complexity**

Linear, O(m + n) where m and n are lengths of both linked lists.

**Memory Complexity**

Constant, O(1)

# Merge Sub Lists in Same List

• We essentially play follow the leader.

• temp goes to the next node you want in line

• h1->h2->temp

• So h1.next = h2, then we follow, so h1 = h2, then we follow again, h2 = temp.

• Now the next node we want in line can be anything, but we choose h1.next to be it

ListNode p = null;

ListNode c = head;

while(c != null)

{

ListNode n = c.next;

c.next = p;

p = c;

c = n;

}

return p;