0002\_Easy\_MergeTwoSortedLists\_#21\_Breakdown

Problem:

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

Merge the two lists in a 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.

Examples:

**Example 1:**

A picture containing clipart, clock

Description automatically generated

**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]

Observations:

* The two lists given are sorted in ascending order.

What needs to be true for this problem to work:

* The two lists need to be merged together to form one in order (ascending) list.

How would a person solve this problem:

* Compare the first node of each list, whichever node is the smallest starts the new list (if they’re equal either one is fine). Pop the node that was chosen and then compare the next node with the node of the other list. Repeat.

Brute force:

* Iterate through each list simultaneously comparing each node, the node that is less than is added to the merged list.
* Time Complexity: O(n + k), both lists must be iterated through.
  + Specifically: O(n + k), if one lists is significantly shorter than the other, the rest of the longer list can be appended to the merged list as there are no other nodes to compare to the smaller list, thus avoiding iterating through the rest of the longer list.
* Space Complexity: O(1), only a few pointers are used.

Optimize (BUD, bottlenecks, unnecessary code, duplicate code):

* The brute force solution is optimal.

Pseudocode:

* Create a node to mark the head of the list.
* Create a dummy node to start the merged list.
* While list 1 and list 2 are not finished,
  + If the current node of list 1 is less than the current node of list 2,
    - Add the current node of list 1 to the merged list.
    - Iterate to the next node in list 1.
  + Else (the current node of list 2 is less than the current node of list 1),
    - Add the current node of list 2 to the merged list.
    - Iterate to the next node in list 2.
  + Iterated to the next node in the merged list.
* Add the remaining nodes in the longer list to the merged list.
* Return the head of the merged list.

Recursive:

* Writing the code as recursive increases the Space Complexity to O(n + k) due to the call stack calling each element of each list.

Pseudocode:

* Define the base case as when list 1 is null,
  + Return list 2.
* Or list 2 is null
  + Return list 1.
* If the current node of list 1 is less than the current node of list 2,
  + The next node of list 1 is the call of the mergeTwoLists function iterating to the next node in list 1,
  + return the current node of list 1.
* Else
  + The next node of list 2 is the call of the mergeTwoLists function iterating to the next node in list 2,
  + Return the current node of list 2.