Sort a linked list of 0s, 1s and 2s

Given a linked list of **0s, 1s and 2s**, The task is to sort the list in **non-decreasing** order.

Examples:

Input: *1 -> 1 -> 2 -> 0 -> 2 -> 0 -> 1 -> NULL*
Output: *0 -> 0 -> 1 -> 1 -> 1 -> 2 -> 2 -> NULL*

Input: *1 -> 1 -> 2 -> 1 -> 0 -> NULL*
Output: *0 -> 1 -> 1 -> 1 -> 2 -> NULL*

[Expected Approach – 1] By Maintaining Frequency – O(n) Time and O(1) Space:

*The idea is to traverse the linked List and **count** the number of nodes having values **0, 1 and 2** and store them in an **array** of size 3, say **cnt[]** such that*

- ***cnt[0]** = count of nodes with value 0*
- ***cnt[1]** = count of nodes with value 1*
- ***cnt[2]** = count of nodes with value 2*

*Now, traverse the linked list again to fill the first **cnt[0]** nodes with **0**, then next **cnt[1]** nodes with **1** and finally **cnt[2**] nodes with **2**.*