Thursday, May 1, 2025 5:36 PM

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

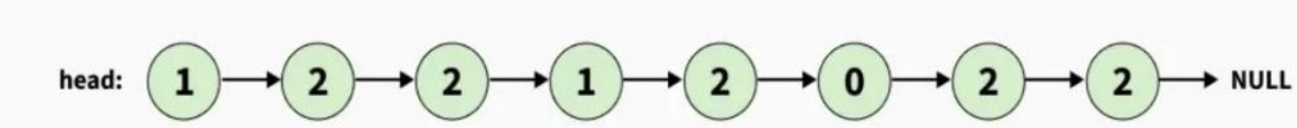
Sort a linked list of 0s, 1s and 2s

Difficulty: **Medium** Accuracy: **60.75%** Submissions: **246K+** Points: **4** Average Time: **30m**

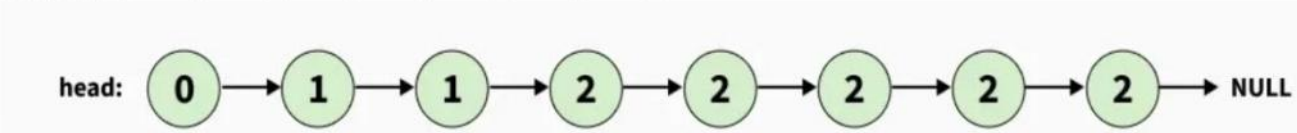
Given the **head** of a linked list where nodes can contain values **0s**, **1s,** and **2s** only. Your task is to **rearrange** the list so that all **0s** appear at the beginning, followed by all **1s**, and all **2s** are placed at the end.

Examples:

Input: head = 1 → 2 → 2 → 1 → 2 → 0 → 2 → 2

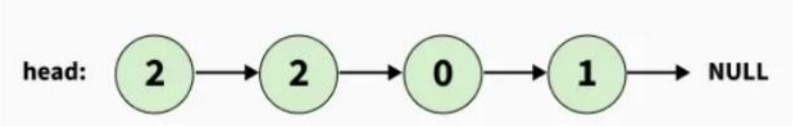


Output: 0 → 1 → 1 → 2 → 2 → 2 → 2 → 2

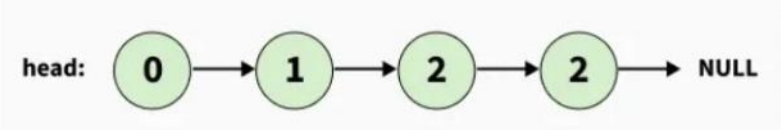


Explanation: All the 0s are segregated to the left end of the linked list, 2s to the right end of the list, and 1s in between.

Input: head = 2 → 2 → 0 → 1



Output: 0 → 1 → 2 → 2



Explanation: After arranging all the 0s, 1s and 2s in the given format, the output will be 0 → 1 → 2 → 2.

[Expected Approach - 2] By updating links of Nodes -

O(n) Time & O(1) space:-

The idea is to maintain 3 pointers named zero, one and two to point to current ending nodes of linked lists containing 0, 1 and 2 respectively. For every traversed node, we attach it to the end of its corresponding list.

- If the current node's value is 0, append it after pointer zero & move zero to current node.
- do same for the 1 and 2.

Finally we link all three lists. To avoid many null checks, we use three dummy pointers zeroD, oneD and twoD that work as dummy headers of three lists.

```
};*/
class Solution {
  public:
    Node* segregate(Node* head) {
        // code here
        if(!head || !(head->next))return head;

        // Create three dummy nodes to point to the beginning of
        // three linked lists. These dummy nodes are created to avoid
        // null checks
        Node* zeroD = new Node(0);
        Node* oneD = new Node(0);
        Node* twoD = new Node(0);

        // Traverse the list
        Node* curr = head;
        while(curr != NULL){
            if(curr ->data == 0){
                // If the data of the current node is 0,
                // append it pointer zero and update zero
                zero->next = curr;
                zero = zero->next;
            }
            else if(curr->data == 1){
                // If the data of the current node is 1
                // append it to pointer one and update one
                one->next = curr;
                one = one->next;
            }
            else{
                // If the data of the current node is 2,
                // append it to pointer two and update two
                two->next = curr;
                two = two->next;
            }
            curr = curr->next;
        }
        // Combine the three lists
        if(oneD->next)zero->next = oneD->next;
        else zero->next = twoD->next;

        // Updated head
        head = zeroD->next;
        return head;
    }
};
```