SORTING A LINKED LIST

☆ In this problem, we are given a linked list and our job is to sort it.

Bruteforce solution is to obviously store all the elements of the LinkedList in an array, sort the elements and place them back. Time Complexity is $O(2N + N\log N)$ while Space Complexity $O(N)$.

Optimal Solution is to implement merge sort within the linked list. We start by breaking the linkedlist into halves until we have single element linked lists. Then we merge each list in a sorted manner.

C++:
```
Node* mergeSort(Node * head) {
    if(head == nullptr || head->next ==
                                  nullptr) {
        return head;
    }
    Node * middle = findMiddle(head);
    Node * lH = head;
    Node * rH = middle->next;
    middle->next = nullptr;
    lH = mergeSort(lH);
    rH = mergeSort(rH);
    return  mergeBack (lH, rH) ;
}

Node * mergeBack(Node * l1, Node * l2) {
    Node * dN = new Node(-1) ;
    Node * temp = dN ;
    while(l1 != nullptr && l2 != nullptr) {
        if (l1 -> data < l2 -> data) {
            temp -> next = l1 ;
            temp = l1 ;
            l1 = l1 -> next ;
        } else {
            temp -> next = l2 ;
            temp = l2 ;
            l2 = l2 -> next ;
        }
    }
    if(l1 != nullptr) {
        temp -> next = l1 ;
    } else {
        temp -> next = l2 ;
    }
    return dN -> next ;
}

Node * findMiddle (Node * head) {
    Node* s = head, fast = head ;
    while(f != nullptr && f -> next != nullptr){
        s = s -> next ;
        f = f -> next -> next ;
    }
    return s ;
}
```