Daily Coding Problem #208

Problem

This problem was asked by LinkedIn.

Given a linked list of numbers and a pivot k, partition the linked list so that all nodes less than k come before nodes greater than or equal to k.

For example, given the linked list `5 -> 1 -> 8 -> 0 -> 3` and `k = 3`, the solution could be `1 -> 0 -> 5 -> 8 -> 3`.

Solution

First let's define some helpful classes. In addition to a simple `Node` class, we will create a `LinkedList` class that can both `insert` elements at the front of the list and `append` elements at the back. Note that `append` here is `O(1)`, since we keep track of a tail pointer.

```
class Node:
    def __init__(self, val, next=None):
```

```
        self.val = val
        self.next = next

class LinkedList:
    def __init__(self):
        self.head = None
        self.tail = None

    def insert(self, data):
        if not self.head:
            self.head = self.tail = Node(data)
        else:
            tmp = Node(data)
            tmp.next = self.head
            self.head = tmp

    def append(self, data):
        if not self.head:
            self.head = self.tail = Node(data)
        else:
            tmp = Node(data)
            self.tail.next = tmp
            self.tail = self.tail.next
```

Now, one way we could solve this would be to initialize three linked lists to hold elements smaller than, equal to, and larger than the pivot, respectively. Then when we traverse the input list, we add each element to the appropriate linked list. Finally, we return a concatenation of our lists in the order low, middle, high.

```
def partition(head, pivot):
    low = LinkedList()
    middle = LinkedList()
    high = LinkedList()

    while head:
        if head.val < pivot:
            low.append(head.val)
```

```
        elif head.val == pivot:
            middle.append(head.val)
        else:
            high.append(head.val)
        head = head.next

    m = middle.head
    while m:
        low.append(m.val)
        m = m.next

    h = high.head
    while h:
        low.append(h.val)
        h = h.next

    return low
```

This is a bit cumbersome, however, and we waste some time during concatenation. Note from the problem description that we only need to guarantee that all nodes with values less than k come before all nodes with values greater than k. In other words, k does not have to be in the middle of the list! As a result, we can solve this in a simpler way: as we traverse the input list, we `insert` elements whose value is less than k into our new linked list, and append everything else:

```
def partition(head, pivot):
    new = LinkedList()

    while head:
        if head.val < pivot:
            new.insert(head.val)
        else:
            new.append(head.val)
        head = head.next

    return new
```

Both of these algorithms are `O(N)` in time and space, as we traverse the input once, and transfer the data to (up to three) new linked lists of size `<= N`.
