Daily Coding Problem #78

Problem

This problem was asked by Google.

Given k sorted singly linked lists, write a function to merge all the lists into one sorted singly linked list.

Solution

A brute force solution here might be to gather all the values of the linked lists into one large array, sort the array, and then recreate a linked list with the values from the array. That would look like this:

```
def merge(lists):
    # Combine all nodes into an array
    arr = []
```

```
    for head in lists:
        current = head
        while current:
            arr.append(current.val)
            current = current.next

    new_head = current = Node(-1) # dummy head
    for val in sorted(arr):
        current.next = Node(val)
        current = current.next

    return new_head.next
```

This would take O(KN log KN) time and O(KN) space, where K is the number of lists and N is the number of elements in the largest list.

A better way would be to take advantage of the inherent sortedness of the input lists. We can keep track, using pointers, of where we are at each list, and pick the minimum of all the pointers. Once we've picked one, we can move that pointer up. This would run in O(KN * K) time and O(K) space.

```
def merge(lists):
    new_head = current = Node(-1)
    while all(lst is not None for lst in lists):
        # Get min of all non-None lists
        current_min, i = min((lst.val, i) for i, lst in enumerate(lists) if lst is not None)
        lists[i] = lists[i].next
        current.next = Node(current_min)
        current = current.next
    return new_head.next
```

An even faster way would be to use a heap to keep track of all the pointers instead. Then we can do this in O(KN * logK)

time, since we'll be using the heapsort ordering to figure out the min in O(log K) time instead of O(K) time.

```
def merge(lists):
    new_head = current = Node(-1)
    heap = [(lst.val, i) for i, lst in enumerate(lists)]
    heapq.heapify(heap)
    while heap:
        current_min, i = heapq.heappop(heap)
        # Add next min to merged linked list.
        current.next = Node(current_min)
        current = current.next
        # Add next value to heap.
        if lists[i] is not None:
            heapq.heappush(heap, (lists[i].val, i))
            lists[i] = lists[i].next
    return new_head.next
```