Daily Coding Problem #177

Problem

This problem was asked by Airbnb.

Given a linked list and a positive integer k, rotate the list to the right by k places.

For example, given the linked list `7 -> 7 -> 3 -> 5` and `k = 2`, it should become `3 -> 5 -> 7 -> 7`.

Given the linked list `1 -> 2 -> 3 -> 4 -> 5` and `k = 3`, it should become `3 -> 4 -> 5 -> 1 -> 2`.

Solution

Let's look at the structure of the solution:

`a -> b -> c -> d -> e, k = 2`

becomes

`d -> e -> a -> b -> c`

To generalize this, we take the k last elements (in order) and move them to the front. We fix the pointers up so that the last element points to the old head, and the kth element's next points to null.

We can accomplish this by using fast and slow pointers.

The basic idea is this. First, advance the fast pointer k steps ahead. Then move the fast and slow pointers together until the fast one hits the end first.

However, to handle the case where k is larger than the length of the linked list itself, we first get the length of the linked list first n, and check `k % n` first.

```
def rotate(head, k):
    fast, slow = head, head
```

```
    for _ in range(k):
        fast = fast.next

    while fast.next is not None:
        slow = slow.next
        fast = fast.next

    new_head = slow.next
    fast.next = head
    slow.next = None

    return new_head
```

This takes O(n) time and O(1) space.