Daily Coding Problem #305

Problem

This problem was asked by Amazon.

Given a linked list, remove all consecutive nodes that sum to zero. Print out the remaining nodes.

For example, suppose you are given the input `3 -> 4 -> -7 -> 5 -> -6 -> 6`. In this case, you should first remove `3 -> 4 -> -7`, then `-6 -> 6`, leaving only `5`.

Solution

The key piece of information here is that the nodes that sum to zero must be consecutive. If this were not the case, we would have to keep track of each value seen so far, and calculate all combinations of values. Instead, we can solve this using a two-pointer approach, where we search for a range of values that sum to zero.

Initially both our start and end pointers will be at the head of our linked list. The end node will then traverse the list, adding each successive value to a running sum. If at any point the total becomes zero, we know we should not print these nodes, so we can advance the start pointer past each of them and continue the process.

For a given start node, if we are able to traverse all remaining nodes without summing to zero, that node must be part of our solution. Once we print the value of this node, we

advance the start pointer one node forward and repeat.

```
class LinkedList:
    def __init__(self, data, next=None):
        self.data = data
        self.next = next

def print_nodes(ll):
    start = end = ll

    while start:
        end = start
        total = 0
        skip = False

        while end:

            total += end.data
            if total == 0:
                start = end
                skip = True
                break
            end = end.next

        if not skip:
            print(start.data)

        start = start.next
```

In the worst case, we must move our end pointer across the remainder of the linked list for each potential start node, taking $O(N^2)$ time. At any given time we will only be tracking the two pointers and our running total, so the total space required is $O(1)$.

Privacy Policy

Terms of Service

Press