Leetcode: Plus One Linked List

January 31, 2018

Question

You are given a non-negative integer represented as a non-empty singly linked list , plus one to the integer. You may assume the integer does not contain any leading zero, except the number 0 itself. The digits are stored such that the most significant digit is at the head of the list.

Example:

Input: $1->2->3$ Output: $1->2->4$

Explanation and Algorithm

The best way to think about this algorithm is recursively, because the first thing we need to figure out is if there's a carry number necessary (i.e. if the sum of a number and $1 = 10$, carry=1, else, carry=0). The base cases for this are pretty straightforward. If the list is empty, return 1. Otherwise, check the carry value. If it is 0, the head (front) of the list stays the same.

Hints

1. Think of how to structure the methods when creating the solution. One good approach to start would be one recursive method for finding the carry value as well as the head value. Then another method that returns the new linked list.

2. In the case that the value of 1 added to the head = 10, how should we manipulate 10 in the code to create the output?

3. In the first recursive method I discuss in the first hint, think about how you should account for adding the 1 carried from a 10 to the next number from the head.

Code

```java
public class Solution {

    public ListNode plusOne(ListNode head) {

        if(plusOneRecursive(head) == 0) {

            return head;

        }

        else {

            ListNode newHead = new ListNode(1);

            newHead.next = head;

            return newHead;

        }

    }

    private int plusOneRecursive(ListNode head) {
        if(head == null){
         return 1;
        }

        int carry = plusOneRecursive(head.next);

        if(carry == 0){
         return 0;
        }

        int value = head.val + 1;
        head.val = value % 10;
        return value/10;

    }

}
```

Run time analysis

Runs in $O(n)$, where n is the size of the linked list to which we're adding 1.