Plus One Linked List

Given a non-negative number represented as a singly linked list of digits, plus one to the number.

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
```

Solution 1

```java
public class Solution {
    public ListNode plusOne(ListNode head) {
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode i = dummy;
        ListNode j = dummy;

        while (j.next != null) {
            j = j.next;
            if (j.val != 9) {
                i = j;
            }
        }

        if (j.val != 9) {
            j.val++;
        } else {
            i.val++;
            i = i.next;
            while (i != null) {
                i.val = 0;
                i = i.next;
            }
        }

        if (dummy.val == 0) {
            return dummy.next;
        }

        return dummy;
    }
}
```

- i stands for the most significant digit that is going to be incremented if there exists a carry
- dummy node can handle cases such as "9->9>-9" automatically

written by oceaniatt original link here

Solution 2

If the +1 was already handled without further carry, then the result is the given head node. Otherwise it's a new node (with carry value 1). In other words, a carry-node is created at the end and gets carried towards the front until it has been fully integrated.

Python

```python
def plusOne(self, head):
    if not head:
        return ListNode(1)
    plused = self.plusOne(head.next)
    head.val += plused != head.next
    if head.val <= 9:
        return head
    head.val = 0
    plused.next = head
    return plused
```

C++

```cpp
ListNode* plusOne(ListNode* head) {
    if (!head)
        return new ListNode(1);
    auto plused = plusOne(head->next);
    head->val += plused != head->next;
    if (head->val <= 9)
        return head;
    head->val = 0;
    plused->next = head;
    return plused;
}
```

Java

```java
public ListNode plusOne(ListNode head) {
    if (head == null)
        return new ListNode(1);
    ListNode plused = plusOne(head.next);
    if (plused != head.next)
        head.val++;
    if (head.val <= 9)
        return head;
    head.val = 0;
    plused.next = head;
    return plused;
}
```

written by StefanPochmann original link here

Solution 3

At the first glance, I want to reverse the inputs, add one, then reverse back. But that is too intuitive and I don't think this is an expected solution. Then what kind of alg would adding one in reverse way for list?

Recursion! With recursion, we can visit list in reverse way! So here is my recursive solution.

```
public ListNode plusOne(ListNode head) {
    if( DFS(head) == 0){
        return head;
    }else{
        ListNode newHead = new ListNode(1);
        newHead.next = head;
        return newHead;
    }
}

public int DFS(ListNode head){
    if(head == null) return 1;

    int carry = DFS(head.next);

    if(carry == 0) return 0;

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

written by hpplayer original link here