addLists

2016 年 4 月 6 日

1 链表求和

你有两个用链表代表的整数，其中每个节点包含一个数字。数字存储按照在原来整数中相反的顺序，使得第一个数字位于链表的开头。写出一个函数将两个整数相加，用链表形式返回和。

样例 给出两个链表 3->1->5->null 和 5->9->2->null，返回 8->0->8->null

```
In [6]: # Definition for singly-linked list.
        class ListNode:
            def __init__(self, x, next=None):
                self.val = x
                self.next = next

            def travel(self):
                while self != None:
                    print self.val, '->',
                    self = self.next
                print 'null'

        class Solution:
            # @param l1: the first list
            # @param l2: the second list
            # @return: the sum list of l1 and l2
            def addLists(self, l1, l2):
                # write your code here
                # Firstly, direct result to the head of l1
                result = l1

                # Sencondly, travel l1 and l2 for sum
                while l1 != None:
                    if l2 == None:
```

```
                return result
            else:
                # add l1 and l2 elements in equal index
                sum = l1.val + l2.val
                large = sum/10
                small = sum - large*10
                l1.val = small
                if large > 0:
                    if l1.next != None:
                        l1.next.val = l1.next.val + large
                    else:
                        # new ListNode for l1
                        l1.next = ListNode(large)

            # direct l2's head to the next node no matter l2's next is None or not
            l2 = l2.next
            # add the rest of l2 to l1 when l1'next is None, and finish add operate
            if l1.next == None:
                l1.next = l2
                break
            else:
                l1 = l1.next

        return result

L1 = ListNode(3, ListNode(1, ListNode(5)))
L2 = ListNode(5, ListNode(9, ListNode(2)))
L1.travel()
L2.travel()
#
sol = Solution()
L3 = sol.addLists(L1, L2)
L1.travel()
L2.travel()
L3.travel()
```

```
3 -> 1 -> 5 -> null
5 -> 9 -> 2 -> null
8 -> 0 -> 8 -> null
5 -> 9 -> 2 -> null
```

```
8 -> 0 -> 8 -> null
```