Question 2.5 from CTCI: Sum Lists

September 9, 2017

Question

You have two numbers represented by a linked list where each node contains a single digit. The digits are stored in reverse order, such as the 1's digit is the head of the list. Write a function that adds two numbers and returns the sum as a linked list.
Input: $(2- > 3- > 7) + (4- > 5- > 6)$ reads as $732 + 654$ Output: $1- > 3- > 8- > 6$

Hints

1. Think of regular addition. For example, if we are adding 345 + 56, we first have to add the ones place and if the sum is ¿ 10, carry a 1 over and add that to the tens place. Think of this algorithm when planning out your solution.

2. Now, apply this to linked lists. We are told that the two linked lists have the ones place at the head, tens next, etc. Think of how you can add the first node of the first list, first node of the second list, and the number to carry over (if applicable).

3. What conditions should make a carry applicable? How will you know if you've reached the end of one list but not the other? If so, how will you add that to the result list? Consider all these edge cases.

Code

```
/* Answer 1 (Known as two-pointer method) */

Node prev = null;
Node temp = null;
int carry = 0; //number to carry over if sum of two numbers > 10
```

```
int sum = 0;
Node newList = null; //head of new linked list with result

   while(list1 != null || list2 != null){

      /*to calculate value of next digit of result, we must add, carry,
          the next digit of list1 (if there is a next digit) and the
          next digit of list2 (if there is a next digit)*/

      sum = carry + (first != null ? first.data : 0)+(second != null ?
          second.data)
      carry = (sum > 10) ? 1 : 0;
      sum = sum % 10; //updated if > 10
      temp = new Node(sum); //create new node with sum as its data

         //if result is empty, set temp as resultant list
         if(newList == null){
            newList = temp;
         }else{ //if it is not the first node, set it as the next node
            prev.next = temp;
         }
      prev = temp; //set up for next insertion
         if(first != null){
            first = first.next;
         }
         if(second != null){
            second = second.next;
         }
   }

   if(carry > 0){
      temp.next = new Node(carry);
   }

return newList;
}
```

Big O analysis

O(m + n), where m is the size of list1 and n is the size of list2. Since the result is the sum of the individual nodes of those lists, the result will be of size m+n.