MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

LAB 4 Write Up (Add Two Numbers II)

Aswin Sampath
saswin@uci.edu
53844684

Date: 26/10/2023

Question:

Description Editorial Solutions (4K) Submissions

445. Add Two Numbers II

Medium 5.7K 279

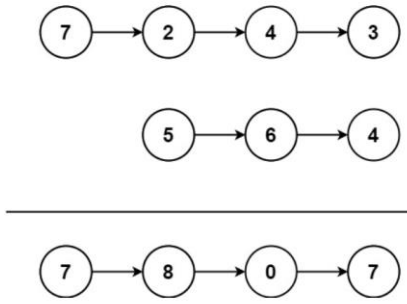
Companies

You are given two **non-empty** linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Description Editorial Solutions (4K) Submissions

Example 1:




```
Input: l1 = [7,2,4,3], l2 = [5,6,4]
Output: [7,8,0,7]
```

Example 2:

```
Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [8,0,7]
```

Example 3:

```
Input: l1 = [0], l2 = [0]
Output: [0]
```

Constraints:

- The number of nodes in each linked list is in the range `[1, 100]`.
- `0 <= Node.val <= 9`

Understanding the Problem:

We are given two linked lists representing non-negative integers. The digits in the linked lists are stored in reverse order, with the most significant digit at the head of the list. We need to add these two numbers and return the result as a new linked list.

Identifying Edge Cases:

- The linked lists could be of different lengths.
- There might be a carry in the most significant digit, causing the result to have an extra digit.
- The linked lists could contain just one digit.

Effective Test Cases:

- Input: l1 = [7,2,4,3], l2 = [5,6,4] Expected Output: [7,8,0,7]
- Input: l1 = [2,4,3], l2 = [5,6,4] Expected Output: [8,0,7]
- Input: l1 = [0], l2 = [0] Expected Output: [0]
- Input: l1 = [9,9,9], l2 = [1] Expected Output: [1,0,0,0]

Algorithmic Solution:

The code uses two stacks to reverse the order of the digits in both linked lists. Then, it iterates through the digits, adding them, and accounting for any carry. The result is stored in a new linked list, which is built in reverse order.

1. Initialize two stacks, s1 and s2, to reverse the digits of l1 and l2.
2. Initialize a carry variable to 0 and a result linked list as nullptr.
3. While either s1 is not empty or s2 is not empty or there is a carry:
    a. Calculate the sum as carry + digit from s1 + digit from s2.
    b. Update carry as the integer division of sum by 10.
    c. Create a new node with the value sum % 10 and make it the new head of the result linked list.
4. Return the result linked list.

Time and Space Complexity Analysis:

Time Complexity: The code iterates through both linked lists once to build the stacks, which takes O(n) time. Then, it iterates through the stacks to calculate the sum, which also takes O(n) time. Overall, the time complexity is **O(n).**

Space Complexity: The code uses two stacks to store the digits, which have a space complexity of O(n). The space used by the result linked list is also O(n). Therefore, the space complexity is O(n).