Documentation for Solution: Add Two Numbers II

Table of Contents

1. Problem Statement

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.

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]

2. Intuition

To solve this problem, we need to:

- Add the corresponding digits of the two numbers.
- Account for the carry that may arise when the sum of the digits exceeds 9.
- Ensure the sum is represented in a linked list format where the most significant digit comes first.

The problem's challenge lies in the fact that the digits are provided from most significant to least significant, while traditional addition typically starts from the least significant digit.

Approach Outline:

a. Reverse the Lists: Reverse both input linked lists so that we can add the digits from the least significant to most significant.
b. Sum the Lists: Traverse both lists, summing the digits one by one, while considering the carry.
c. Reverse the Result: Once the addition is complete, reverse the result to restore the most significant digit at the front.

3. Key Observations

- Linked List Representation: Each linked list node represents a single digit of the number.
- Most Significant Digit First: The digits are stored such that the most significant digit comes first in the list.
- Carry Handling: As we add digits, the sum might exceed 9, in which case we need to propagate the carry to the next position.
- Reversal Requirement: Reversing the lists helps in simplifying the addition process as we can start from the least significant digit.

4. **Approach**

a. Reverse Both Input Lists: To simplify the addition, reverse both linked lists to process the digits from least significant to most significant.
b. Add the Digits: Traverse both reversed linked lists, adding corresponding digits and carrying over any excess (i.e., if the sum is greater than 9).
c. Construct Result List: Start building the result linked list by appending the sum digits, taking care of the carry.
d. Reverse the Result: Since the sum list will be constructed in reverse order, reverse it before returning the final result.

5. **Edge Cases**

- Input Lists are the Same Length: This is a standard case; no special handling is needed.
- One List is Shorter than the Other: The shorter list should be treated as having leading zeros for the purpose of addition.
- Carry Overflow: Ensure that if there's a carry left after the last digit addition, it is added as a new node in the result list.
- Both Lists Represent Zero: If both lists represent zero, the result should also be a single node containing zero.

6. **Complexity Analysis**

Time Complexity

- Reversing the Lists: O(n) for each of the two lists (n is the length of the longer list).
- Adding the Lists: O(n), as we process each node of both lists.
- Reversing the Result: O(n) for the final list.

Thus, the overall time complexity is O(n), where n is the maximum length of the two input lists.

Space Complexity

- The space complexity is O(n), where n is the length of the result list. This is due to the storage needed for the result linked list.

7. **Alternative Approaches**
    a. **Without Reversing the Lists:**

        - Instead of reversing the lists, we can use stacks to store the digits of each list. After pushing all the digits of both lists onto their respective stacks, we can pop them and add them digit by digit, handling the carry. This method also avoids reversing the linked lists but requires additional space for the stacks.

    b. **Using a Linked List with Stack:**

        - Push all the digits of both lists onto stacks.
        - Pop the digits and sum them up, while managing the carry.
        - Finally, the result can be built by pushing the sum digits onto a new stack, then converting it to a linked list.

    However, this approach would still require O(n) space and additional operations compared to the current method.

8. **Test Cases**

    Test Case 1:

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

    Test Case 2:

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

    Test Case 3:

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

Test Case 4:

Input: l1 = [1, 0], l2 = [9, 9, 9]
Output: [1, 0, 0, 0]

9. Final Thoughts

This solution provides an efficient way to solve the problem of adding two numbers represented by linked lists. By reversing the lists, performing the addition in reverse order, and then reversing the result, we ensure that the addition process is straightforward while maintaining the correct order of digits in the output.