LeetCode Answers

Keaton Clark

April 23, 2022

Contents

Part I

Hard

Part II

Medium

Chapter 1

2. Add Two numbers

Description-
You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, 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.
Results-
Runtime: 47 ms, faster than 55.66% of C++ online submissions for Add Two Numbers.
Memory Usage: 71.4 MB, less than 50.94% of C++ online submissions for Add Two Numbers.

Try 1: Worked great until I realized that the numbers can be up to $9 * 10^{100}$. Turn each list into ints, sums them, and uses a recursive function to convert it back into a reversed linked list.

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode *addTwoNumbers(ListNode* l1, ListNode* l2) {
        int place = 1;
        int s1 = 0, s2 = 0;
        while(l1) {
            s1 += l1->val * place;
            place *= 10;
            l1 = l1->next;
        }
        place = 1;
        while(l2) {
            s2 += l2->val * place;
            place *= 10;
            l2 = l2->next;
        }
        int sum = s1 + s2;
        return createNodes(sum, nullptr);
    }
    ListNode *createNodes(int x, ListNode *curr) {
        if (x >= 10)
            curr = createNodes(x / 10, curr);
        return new ListNode(x % 10, curr);
    }
};
```

Try 2: The correct way to do it. Sums them just like an adder in a cpu would. Sums each digit and has a carry digit to keep track of overflows.

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */

class Solution {
public:
    ListNode *addTwoNumbers(ListNode* l1, ListNode* l2) {
        int sum = l1->val + l2->val;
        int carry = (sum > 9) ? 1 : 0;
        auto output = new ListNode();
        auto curr = output;
        l1 = l1->next;
        l2 = l2->next;
        if (carry == 1) {
            curr->val = sum % 10;
        } else {
            curr->val = sum;
        }
        while (l1 && l2) {
            sum = l1->val + l2->val + carry;
            carry = (sum > 9) ? 1 : 0;
            curr->next = new ListNode();
            curr = curr->next;
            if (carry == 1) {
                curr->val = sum % 10;
            } else {
                curr->val = sum;
            }
            l1 = l1->next;
            l2 = l2->next;
        }
        while (l1) {
            sum = l1->val + carry;
            carry = (sum > 9) ? 1 : 0;
            curr->next = new ListNode((carry) ? sum % 10 : sum);
            curr = curr->next;
            l1 = l1->next;
        }
        while (l2) {
            sum = l2->val + carry;
            carry = (sum > 9) ? 1 : 0;
            curr->next = new ListNode((carry) ? sum % 10 : sum);
            curr = curr->next;
            l2 = l2->next;
        }
        if (carry) {
            curr->next = new ListNode(carry);
        }
        return output;
    }

};
```