两个链表做加法\_LeetCode\_445\_II\_AddTwoNumbers\_Medium

\* T1:LeetCode\_2\_I\_AddTwoNumbers\_Medium\_两个链表做加法

\* T2:LeetCode\_445\_II\_AddTwoNumbers\_Medium\_两个链表做加法

# T1:LeetCode\_2\_I\_AddTwoNumbers\_Medium\_两个链表做加法

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*T1:LeetCode\_2\_I\_AddTwoNumbers\_Medium\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* T1:LeetCode\_2\_I\_AddTwoNumbers\_Medium

\* 难度：Medium

\* https://leetcode.com/problems/add-two-numbers/description/

\* DateTime：2018-10-08

\* <p>

\* 题目介绍：

\* 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 contain a single digit.

\* Add the two numbers and return it as a linked list.

\* <p>

\* You may assume the two numbers do not contain any leading zero,

\* except the number 0 itself.

\* <p>

\* Example:

\* Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)

\* Output: 7 -> 0 -> 8

\* Explanation: 342 + 465 = 807.

## 思路分析

\* 思路分析：思想都是一样的，和实现加法一样，carry保存进位值。

\* 实现1、实现2、实现3;第3种实现更好些。

## Java代码

### 方法实现1

public ListNode **addTwoNumbers1**(ListNode head1, ListNode head2) {

if (head1 == null) return head2;

if (head2 == null) return head1;

ListNode dummy = new ListNode(-1);

ListNode curr = dummy;

int sum = 0, carry = 0;

//两者都不为null时

while (head1 != null && head2 != null) {

sum = head1.val + head2.val + carry;

curr.next = new ListNode(sum % 10);

curr = curr.next;

carry = sum / 10;

head1 = head1.next;

head2 = head2.next;

}

//剩下一个不为null或都为null

ListNode after = (head1 == null) ? head2 : head1;

while (after != null) {

sum = after.val + carry;

curr.next = new ListNode(sum % 10);

curr = curr.next;

after = after.next;

carry = sum / 10;

}

//最后处理进位

curr.next = (carry == 0) ? null : new ListNode(carry);

return dummy.next;

}

### 方法实现2

\* 两个链表一起处理

public ListNode addTwoNumbers2(ListNode head1, ListNode head2) {

ListNode dummy = new ListNode(-1);

ListNode curr = dummy;

int sum = 0, carry = 0;

while (head1 != null || head2 != null) {

int x = (head1 == null) ? 0 : head1.val;

int y = (head2 == null) ? 0 : head2.val;

sum = x + y + carry;

carry = sum / 10;

curr.next = new ListNode(sum % 10);

curr = curr.next;

if (head1 != null) head1 = head1.next;

if (head2 != null) head2 = head2.next;

}

if (carry != 0)

curr.next = new ListNode(carry);

return dummy.next;

}

### 代码实现3：这种比较好些

\* 两个链表一起处理

public ListNode addTwoNumbers(ListNode head1, ListNode head2) {

ListNode dummy = new ListNode(-1);

ListNode curr = dummy;

int sum = 0, carry = 0;

while (head1 != null || head2 != null) {

sum = carry;

if (head1 != null) {

sum += head1.val;

head1 = head1.next;

}

if (head2 != null) {

sum += head2.val;

head2 = head2.next;

}

carry = sum / 10;

curr.next = new ListNode(sum % 10);

curr = curr.next;

}

if (carry != 0) curr.next = new ListNode(carry);

return dummy.next;

}

# T2:LeetCode\_445\_II\_AddTwoNumbers\_Medium\_两个链表做加法

## 题目介绍

\* LeetCode\_2\_I\_445\_II\_AddTwoNumbers\_Medium

\* 难度:Medium

\* https://leetcode.com/problems/add-two-numbers-ii/description/

\* DateTime: 2018-10-08 16:28

\* <p>

\* 题目介绍：

\* You are given two non-empty linked lists

\* representing two non-negative integers.

\* The most significant digit comes first and each of their nodes

\* contain a single digit. Add the two numbers and return it as a linked list.

\* You may assume the two numbers do not contain any leading zero,

\* except the number 0 itself.

\* Follow up:

\* What if you cannot modify the input lists? In other words,

\* reversing the lists is not allowed.

\* <p>

\* Example:

\* Input: (7 -> 2 -> 4 -> 3) + (5 -> 6 -> 4)

\* Output: 7 -> 8 -> 0 -> 7

\* <p>

## 思路分析

\* 思路分析：**栈+节点插入法**。

\* 该题与I的区别是完全逆过来，借助栈即可顺利解决。

\* 难点是：输出链表的顺序问题。

\* 按照正常情况，加法之后的链表为逆序的，然后需要逆置；

\* 这里采用临时头节点dummy，在做加法过程中，

\* 不断在dummy后面插入元素,从而实现逆序增长。

## Java代码

/\*\*

\* 栈+节点插入法

\*/

public ListNode addTwoNumbersII(ListNode head1, ListNode head2) {

//特殊情况判断

if (head1 == null) return head2;

if (head2 == null) return head1;

//将链表元素值压入栈

Deque<Integer> stack1 = new ArrayDeque<Integer>();

Deque<Integer> stack2 = new ArrayDeque<Integer>();

while (head1 != null) {

stack1.push(head1.val);

head1 = head1.next;

}

while (head2 != null) {

stack2.push(head2.val);

head2 = head2.next;

}

//从栈中取数，并做加法运算

int sum = 0, carry = 0;

ListNode dummy = new ListNode(0);//傀儡头节点

while (!stack1.isEmpty() || !stack2.isEmpty()) {//只要有一个不空

sum = carry;

if (!stack1.isEmpty()) sum += stack1.pop();

if (!stack2.isEmpty()) sum += stack2.pop();

carry = sum / 10;

ListNode curr = new ListNode(sum % 10);//待插入的节点

curr.next = dummy.next;

dummy.next = curr;

}

dummy.val = carry;

return (carry == 0)?dummy.next:dummy;

}