



问题描述

<https://leetcode-cn.com/problems/add-two-numbers-ii/>

445. 两数相加 II

难度 中等  171  收藏  分享  切换为英文  关注  反馈

给你两个 **非空** 链表来代表两个非负整数。数字最高位位于链表开始位置。它们的每个节点只存储一位数字。将这两数相加会返回一个新的链表。

你可以假设除了数字 0 之外，这两个数字都不会以零开头。

进阶：

如果输入链表不能修改该如何处理？换句话说，你不能对列表中的节点进行翻转。

示例：

输入：(7 -> 2 -> 4 -> 3) + (5 -> 6 -> 4)
输出：7 -> 8 -> 0 -> 7

通过次数 26,373 | 提交次数 46,538

在真实的面试中遇到过这道题？

是

否

解题思路

自己没想到使用栈，直接反转链表进行实现的。中间还有不少坑，要注意tmp的灵活使用。

代码实现

```
#include<iostream>
#include<stdio.h>
#include<vector>
#include<queue>
#include<string>
using namespace std;

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

void add(ListNode *L, int a)
{
    ListNode *tmp;
    tmp = new ListNode(a);
    L->next = tmp;
```

```

}
int numbit(ListNode *l1)
{
    int num = 0;
    while (l1 != NULL)
    {
        num++;
        l1 = l1->next;
    }
    return num;
}
ListNode* reverse(ListNode *L)
{
    ListNode *head, *tmp;
    head = new ListNode(0);
    tmp = new ListNode(0);
    head->val = -1;
    head->next = L;
    tmp = L->next;
    while (L->next != NULL)
    {
        tmp = L->next;
        L->next = tmp->next;
        tmp->next = head->next;
        head->next = tmp;
    }
    return head->next;
}

void print(ListNode *L)
{
    while (L != NULL)
    {
        cout << L->val<<',';
        L = L->next;
    }
}

class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        int n1 = 0, n2 = 0;
        n1 = numbit(l1);
        n2 = numbit(l2);
        if (n1 >= n2)
        {
            l1 = reverse(l1);
            l2 = reverse(l2);
        }
        else
        {
            ListNode *tmp_1,*tmp_2;
            tmp_1 = reverse(l1);
            tmp_2 = reverse(l2);
            l1 = tmp_2;
            l2 = tmp_1;
        }
    }
};

```

```

    }
    ListNode *head;
    head = l1;
    while (l2 != NULL)
    {
        l1->val += l2->val;
        if (l1->val >= 10)          //进行进位
        {
            ListNode *addbit;
            addbit = l1;
            while (addbit->val >= 10)
            {
                if (addbit->next == NULL)          //假如最高位本来是NULL，新建一个
                节点
                {
                    ListNode *tmp;
                    tmp = new ListNode(0);
                    addbit->next = tmp;
                }
                addbit->next->val++;
                生进位为止
                addbit->val -= 10;
                addbit = addbit->next;
            }
            l1 = l1->next;
            l2 = l2->next;
        }
        return reverse(head);
    }
};

```

```

int main()
{
    ListNode *L1,*tmp,*L2;
    L1 = new ListNode(1);
    L2 = new ListNode(9);
    add(L2, 9);

    /*
    L1->val = 1;
    L1->next = NULL;
    tmp = new ListNode(0);
    tmp->val = 2;
    tmp->next = NULL;
    L1->next = tmp;

    tmp = new ListNode(3);
    L1->next->next = tmp;

    print(L2);
    L2 = reverse(L2);
    print(L2);
    */
    Solution s;
    L1=s.addTwoNumbers(L1, L2);
}

```

```

print(L1);
return 0;
}

```

下用Python实现栈的写法：Python中没有stack,但是list可以用append 和 pop完成栈的功能

```

class Solution:
    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:
        s1,s2=[],[]
        while l1:
            s1.append(l1.val)      # 入栈
            l1=l1.next
        while l2:
            s2.append(l2.val)      # 入栈
            l2 = l2.next
        ans = None
        carry = 0                  # 进位
        while s1 or s2 or carry != 0:
            a = 0 if not s1 else s1.pop()    # 当s1不空时, not s1 是false, s1pop
            b = 0 if not s2 else s2.pop()
            cur = a+ b + carry
            carry = cur // 10
            cur %= 10
            curnode = ListNode(cur)          # 头插法创建新链表
            curnode.next=ans
            ans = curnode
        return ans

```

前面自己写的效果不够好，再写个栈实现的看能不能时间快一点：

```

class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        stack<int> s1;
        stack<int> s2;
        while (l1 != NULL)
        {
            s1.push(l1->val);
            l1 = l1->next;
        }          //两链表入栈
        while (l2 != NULL)
        {
            s2.push(l2->val);
            l2 = l2->next;
        }
        int carry = 0;
        ListNode *head = NULL;
        while (!s1.empty() || !s2.empty() || carry > 0)    //'或' 是因为结果保存在
        //新链表，三者有一个不是空就需要继续计算
        {
            int sum = carry;
            sum += s1.empty() ? 0 : s1.top();              //不能pop,pop不返回值
            sum += s2.empty() ? 0 : s2.top();

```

```
        if (!s1.empty())
            s1.pop();
        if (!s2.empty())
            s2.pop();
        ListNode *tmp = new ListNode(sum % 10);
        tmp->next = head;
        head = tmp;
        carry = sum / 10;
    }
    return head;
};
```