拉勾教育
—互联网人实战大学—

# 链表+数学：两数相加

中等/链表、数学思维

# 学习目标

拉勾教育
—互联网人实战大学—

- 了解算法题的解题思路
- 链表的特点
- 数学思维的应用

# 题目描述



给出两个非空链表用来表示两个非负整数

其中它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储一位数字

如果我们将这两个数相加起来，则会返回一个新的链表来表示它们的和

您可以假设除了数字 0 之外，这两个数都不会以 0 开头

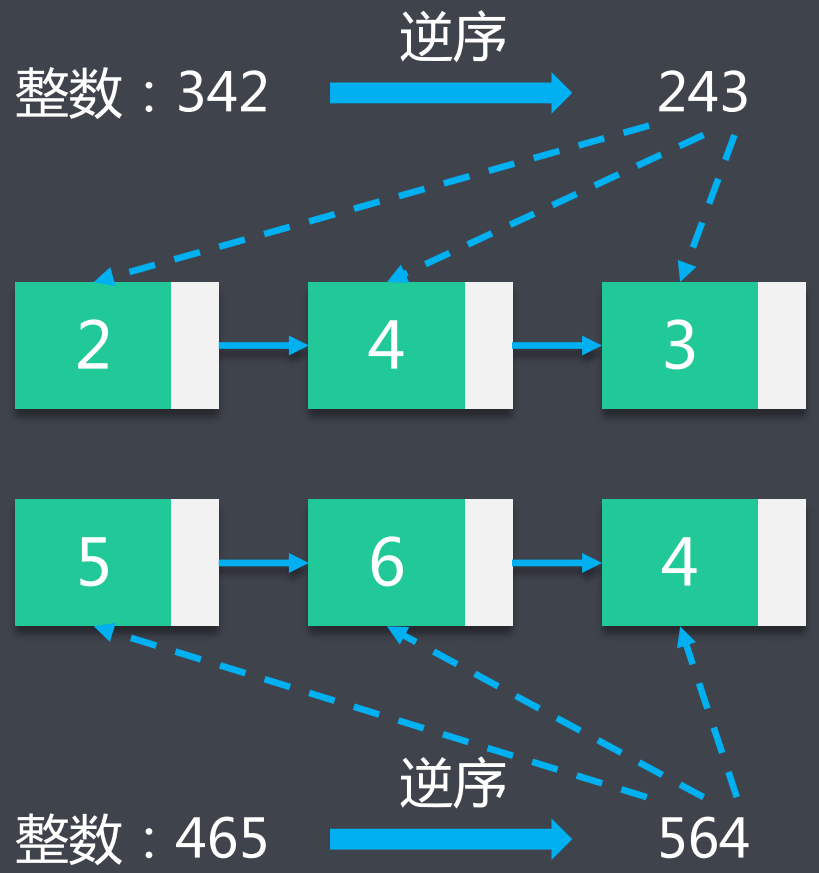
```
输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)
输出：7 -> 0 -> 8
原因：342 + 465 = 807
```

# 一. Comprehend 理解题意



## 题目主干要求

- 一个链表表示一个整数
- 每个节点存储一个数字
- 各自的位数逆序存储
- 链表相加，即链表表示的整数相加
- 返回链表表示的两数之和

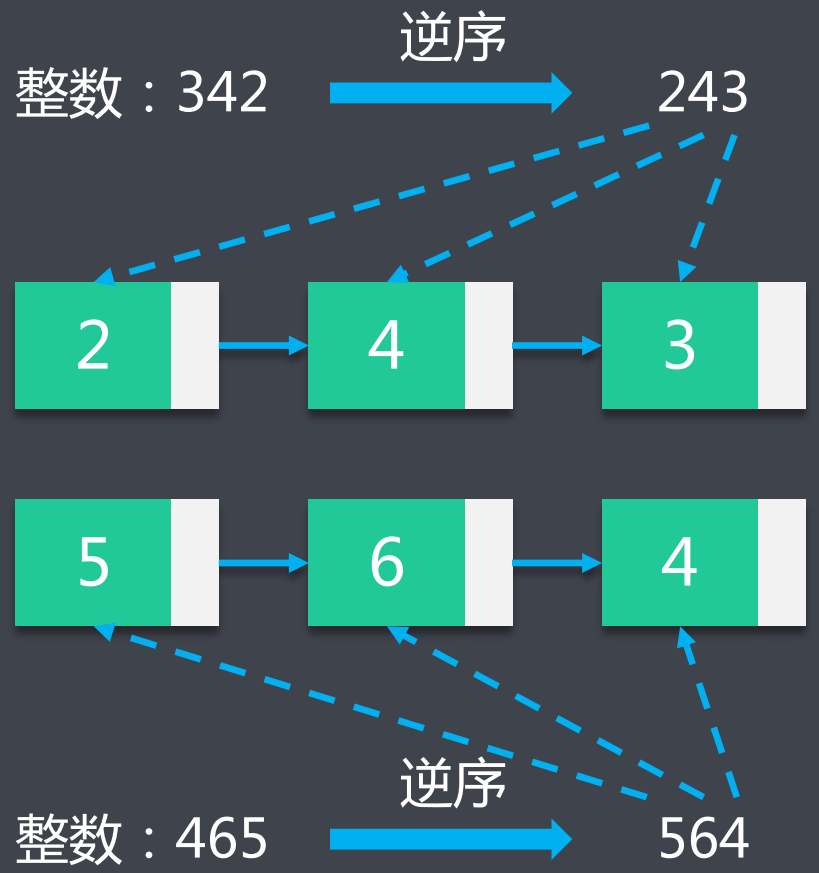


# 一. Comprehend 理解题意



## 细节问题

- 两个链表都是非空的
- 两个非负整数
- 两个数字都不会以0开头
- 数字逆序排列，第一个节点表示整数的最低位
- 题目没有明确整数的范围，这是个容易忽略的点

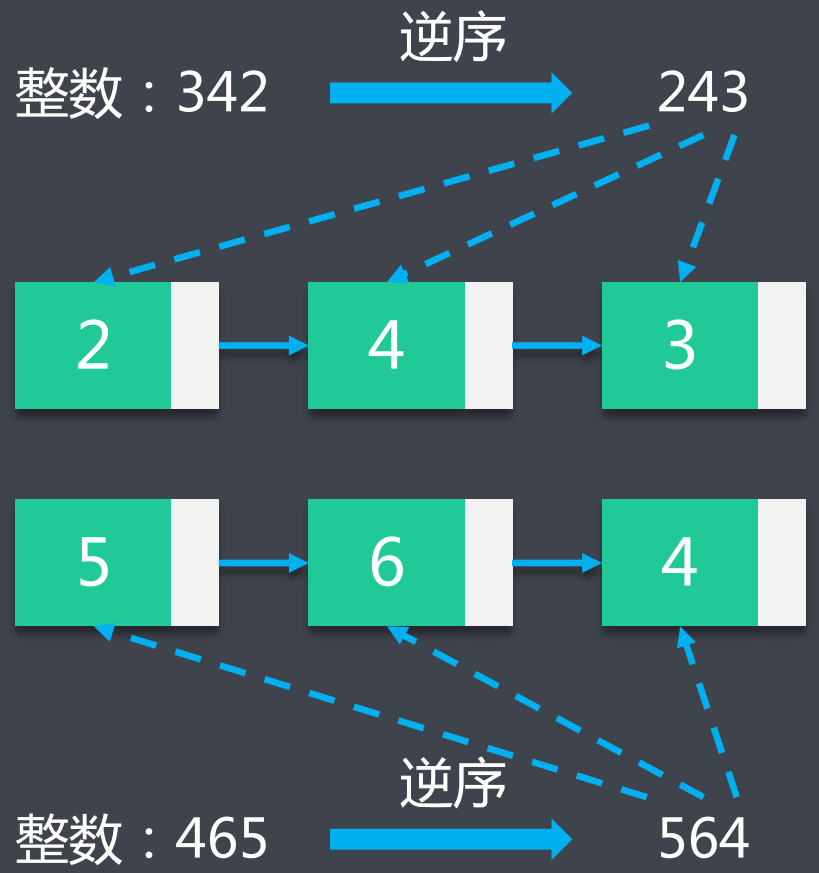


# 一. Comprehend 理解题意

拉勾教育
—互联网人实战大学—

## 解题思路

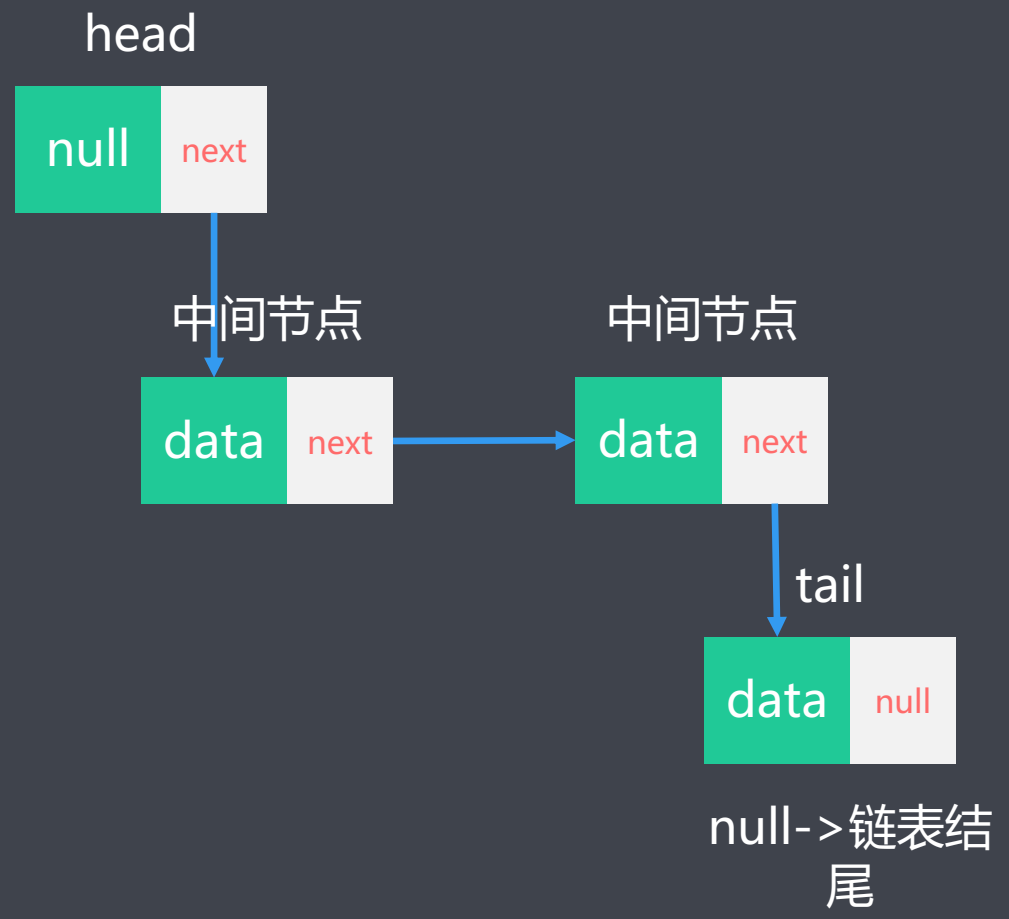
- 解法一：分别将链表转成数字，再相加
- 解法二：直接将对应位置数字相加



# 二. Choose 数据结构及算法思维选择




- 关键知识点：链表
  - 一种物理存储上非连续、非顺序的存储结构
  - 由一系列节点组成，可以在运行时动态生成
  - 每个结点包括两个部分：数据域；指针域
- 链表的特点
  - 数据存储不要求连续空间，不限制容量
  - 数据的逻辑顺序通过指针链接次序实现
  - 从链表头部依次访问后面的节点
  - 在链表表头插入数据的时间复杂度是O(1)



# 二. Choose 数据结构及算法思维选择




- 关键知识点：链表
  - 一种物理存储上非连续、非顺序的存储结构
  - 由一系列节点组成，可以在运行时动态生成
  - 每个结点包括两个部分：数据域；指针域
- 链表的特点
  - 数据存储不要求连续空间，不限制容量
  - 数据的逻辑顺序通过指针链接次序实现
  - 从链表头部依次访问后面的节点
  - 在链表表头插入数据的时间复杂度是O(1)

```
class ListNode {
    int val;
    ListNode next;

    ListNode() {};
    ListNode(int x) {
        val = x;
    }
}
```

# 二. Choose 数据结构及算法思维选择

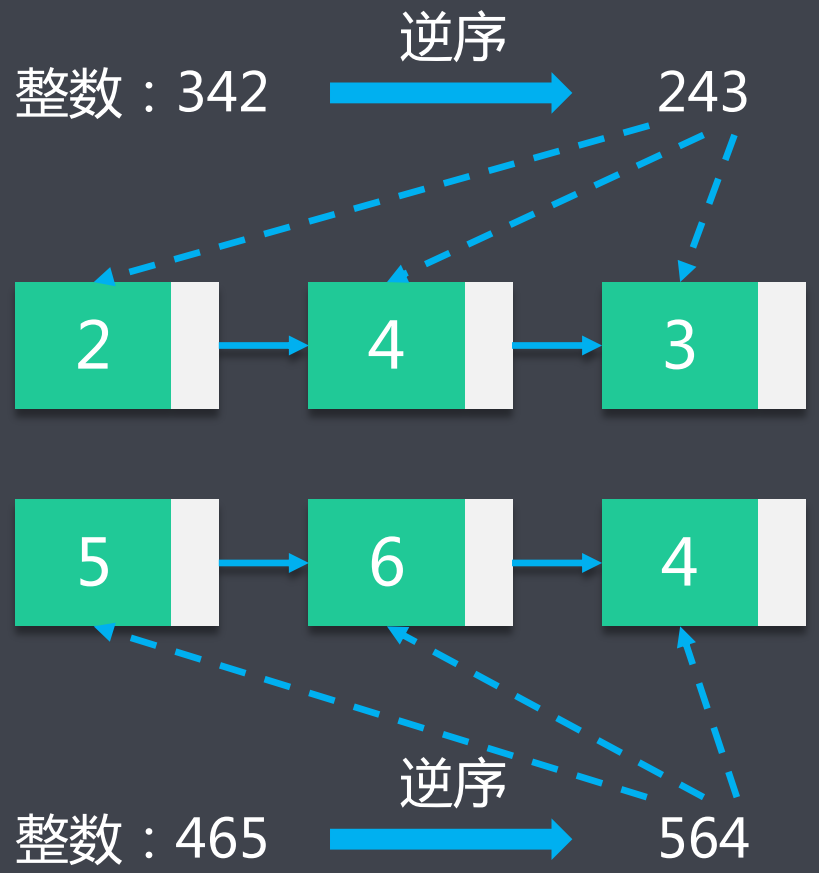
拉勾教育
—互联网人实战大学—

## 数据结构选择

- 链表

## • 算法思维选择

- 解法一：分别将链表转成数字，再相加
  - 遍历
- 解法二：直接将对应位置数字相加
  - 数学

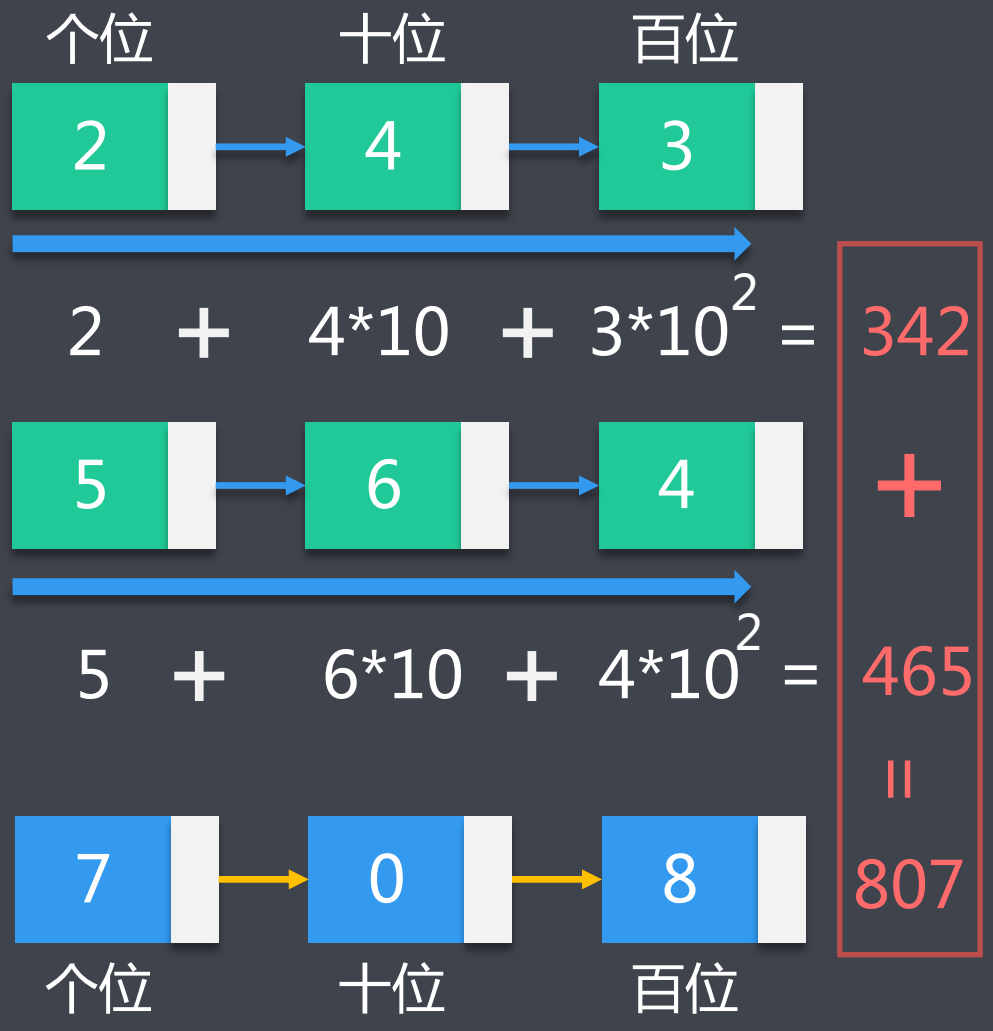


# 三. Code 基本解法及编码实现



## 解法一：暴力解法

1. 遍历两个链表使用数学思维分别将他们转成整数
2. 对两个整数进行求和得到sum
3. 将sum按照数学思维再转成链表



# 三. Code 基本解法及编码实现



## 解法一：暴力解法边界和细节问题

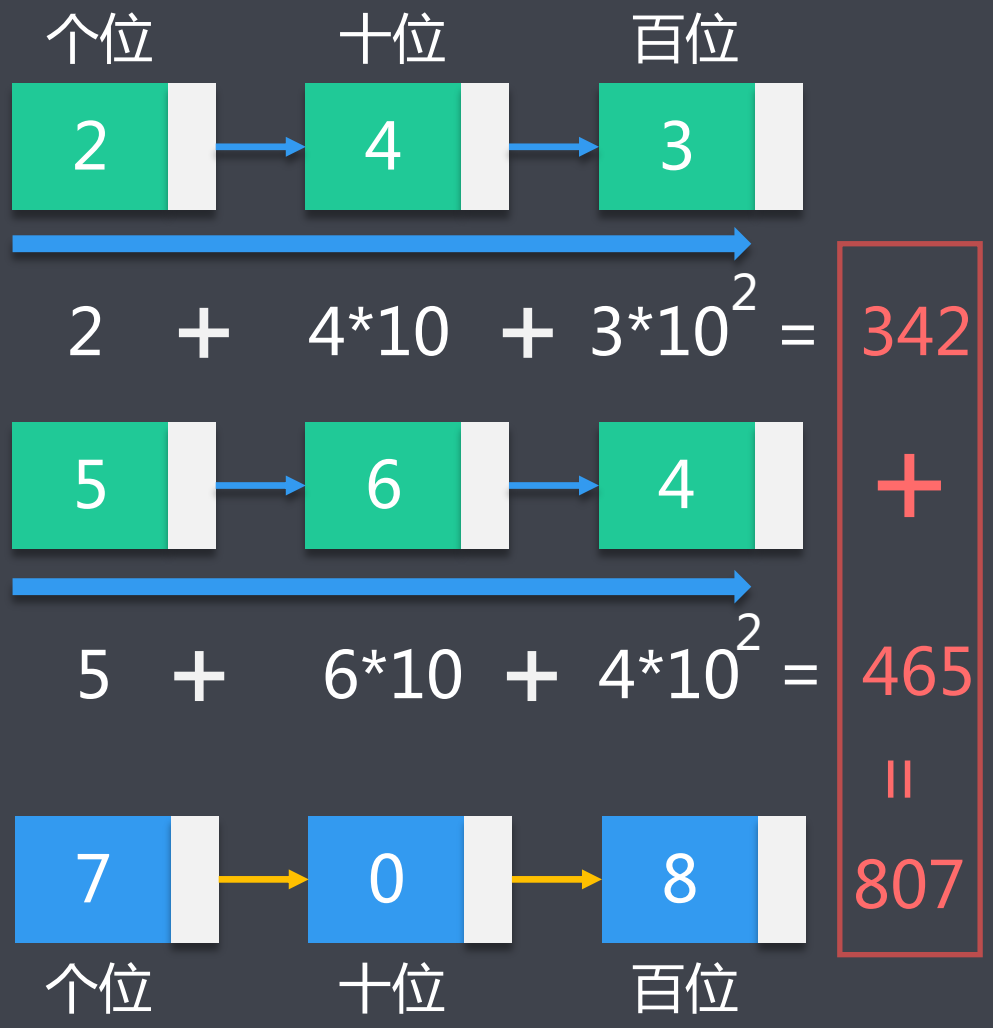
### 边界问题

- 链表转整数时，next==null结尾
- 整数转链表，处理完最高位：value==0

### 细节问题

- 不同节点代表不同的位数
- 题目没有明确整数范围：

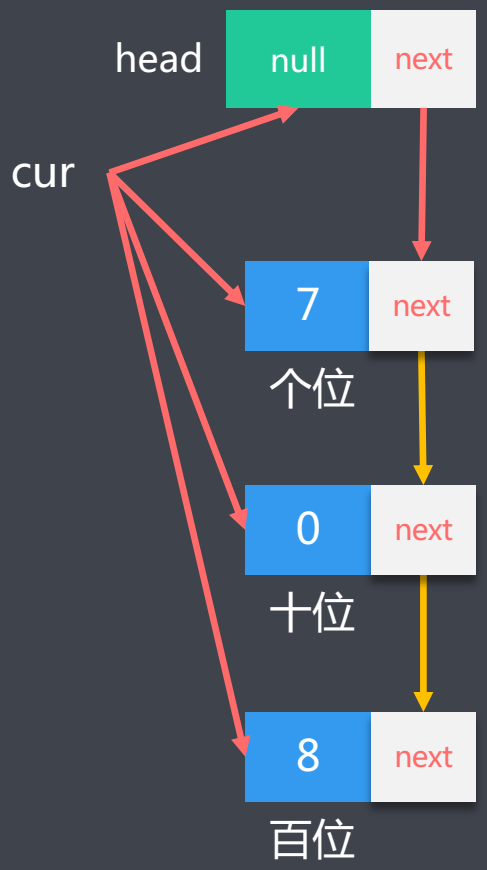
  int 溢出可以用 long，long 溢出？



# 三. Code 基本解法及编码实现



```java
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    long l1Value = 0; // 累加：把链表转成数字，注意次序为逆序
    int digit = 0; // 位数：0代表个位，1代表十位，以此递增
    while (l1 != null) { // 链表的下一个节点，若没有节点返回l1为null
        int pow = (int) Math.pow(10, digit); // 该数字对应的位数，从0开始
        l1Value += (long)l1.val * pow; // 累加：在当前数值基础上增加新的一个高位
        digit++; // 位数加1
        l1 = l1.next; // 链表指向下一个节点
    }
    long l2Value = 0; // 如法炮制
    // ...
    ListNode head = new ListNode(); // 创建一个新链表，头部为空节点
    ListNode cur = head;
    long sum = l1Value + l2Value; // 数字相加，主意越界问题
    if (sum == 0) {
        head = new ListNode(0);
        return head;
    }
    while (sum > 0) { // 数字再转成链表
        int val = (int)(sum % 10); //每次取当前最低位
        cur.next = new ListNode(val); // 创建新节点，插入链表尾部
        cur = cur.next; // 链表尾部指针移动
        sum = sum / 10; // 移除最低位
    }
    return head.next;
}
```



# 三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

```java
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    long l1Value = 0; // 累加：把链表转成数字，注意次序为逆序
    int digit = 0; // 位数：0代表个位，1代表十位，以此递增
    while (l1 != null) { // 链表的下一个节点，若没有节点返回l1为null
        int pow = (int) Math.pow(10, digit); // 该数字对应的位数，从0开始
        l1Value += (long)l1.val * pow; // 累加：在当前数值基础上增加新的一个高位
        digit++; // 位数加1
        l1 = l1.next; // 链表指向下一个节点
    }
    long l2Value = 0; // 如法炮制
    // ...
    ListNode head = new ListNode(); // 创建一个新链表，头部为空节点
    ListNode cur = head;
    long sum = l1Value + l2Value; // 数字相加，主意越界问题
    if (sum == 0) {
        head = new ListNode(0);
        return head;
    }
    while (sum > 0) { // 数字再转成链表
        int val = (int)(sum % 10); //每次取当前最低位
        cur.next = new ListNode(val); // 创建新节点，插入链表尾部
        cur = cur.next; // 链表尾部指针移动
        sum = sum / 10; // 移除最低位
    }
    return head.next;
}
```

## 时间复杂度：O(m+n)

- 分别遍历两个链表：O(m + n)
- 将两数字的和转成链表，在链表尾部插入节点：
  O(k)=O(max(m, n))
  PS: k=max(m, n)或 k=max(m, n)+1
- 总复杂度：O(m+n+max(m,n)) 近似于O(m+n)

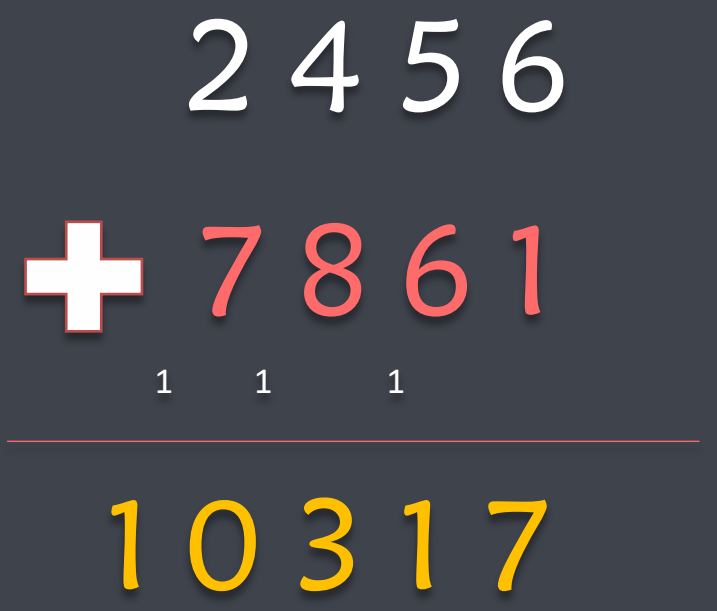
## 空间复杂度：O(max(m,n))

- 创建一个新链表，其长度取决于两数之和的位数：
  O(max(m, n))

# 四. Consider 思考更优解

拉勾教育
—互联网人实战大学—

1. 剔除无效代码或优化空间消耗
   - 减少循环层级和次数
   - 优化代码
2. 寻找更好的算法思维
   - 能否跳过链表转整数的过程？
   - 可否直接将链表对应位置数字相加？
   - 借鉴其它算法



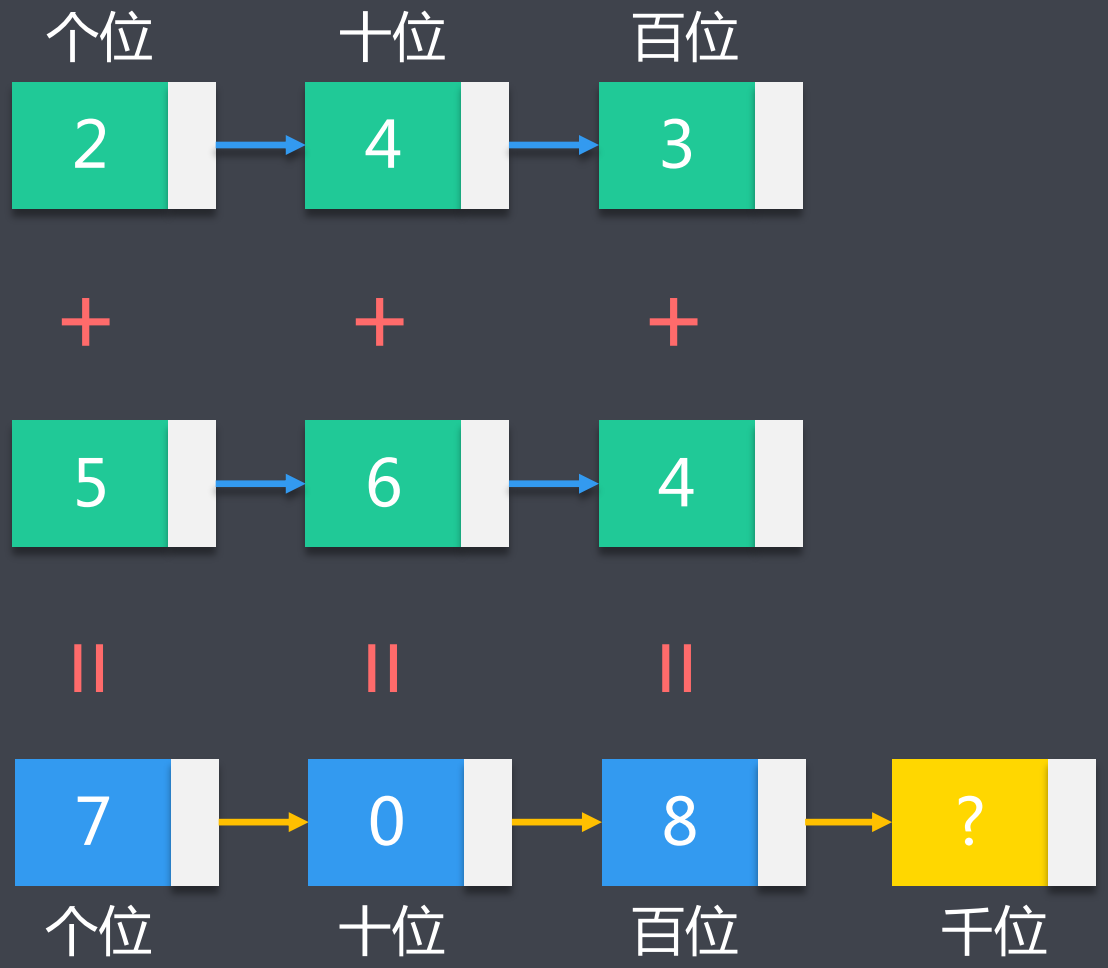
# 五. Code 最优解思路及编码实现



## 最优解：数学思维解法

1. 遍历两个链表
2. 对应位置的节点数值相加
3. 将结果插入新链表尾部

   大于10，则进位，将进位加到下个节点



# 五. Code 最优解思路及编码实现



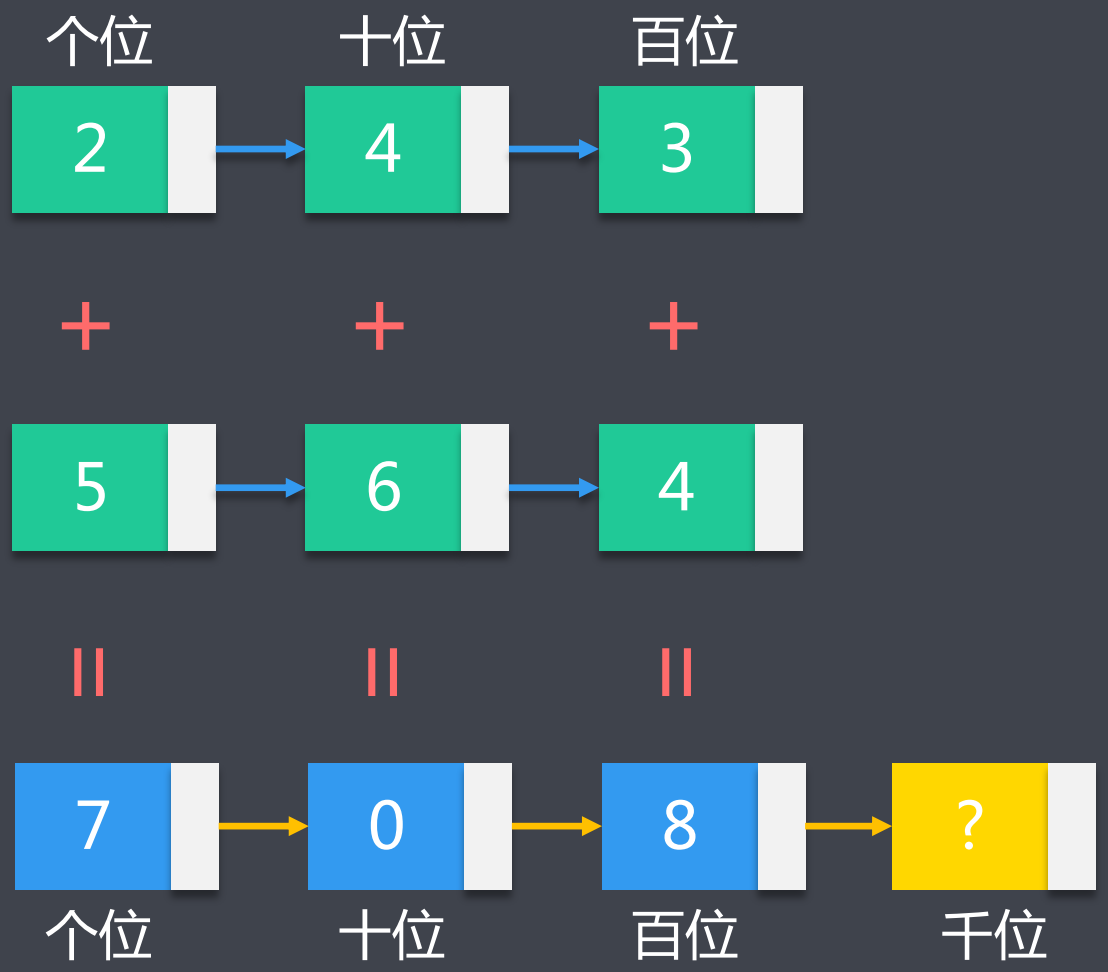
## 最优解：边界和细节问题

### 边界问题

- 两个链表边界：next==null

### 细节问题

- 两个链表长度不一致，短链表高位视为0
- 链表最高位发生进位，结果链表需要增加一个节点存放进位数字



# 五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

```java
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    ListNode p = l1, q = l2; // 原链表的两个遍历指针
    ListNode resultHead = new ListNode(-1); // 结果链表的头结点head
    ListNode curr = resultHead; // 结果链表的遍历指针，代表当前操作的节点
    int carry = 0; // 进位
    // 1.遍历两个链表
    while (p != null || q != null) { // 以较长的链表为准
        // 获取当前节点值：链表较短，已无节点，取0
        int x = (p != null) ? p.val : 0;
        int y = (q != null) ? q.val : 0;
        // 2.对应位置的节点数值相加
        int sum = carry + x + y;
        carry = sum / 10; // 如何得到进位：和对10求整，得到此次计算的进位
        int num = sum % 10; // 存放到新链表节点中的数值
        // 3.将计算结果插入新链表尾部
        curr.next = new ListNode(num); // 创建新节点，追加到结果链表的尾部
        curr = curr.next; // 结果链表的当前节点向后移动
        // 循环的迭代部分：原链表的两个遍历指针
        p = p == null ? p : p.next;
        q = q == null ? q : q.next;
    }
    if (carry > 0) { // 处理进位节点
        curr.next = new ListNode(carry);
    }
    return resultHead.next;
}
```

## 时间复杂度：O(max(m,n))

- 两个链表的加法运算：O(max(m,n))
- 与暴力解法的O(m+n)差别不大，但大数运算以及循环次数来看，更优

## 空间复杂度：O(max(m,n))

- 创建需要创建一个新链表：O(max(m, n))

执行耗时:2 ms,击败了99.93% 的Java用户
内存消耗:38.5 MB,击败了98.99% 的Java用户

# 六. Change 变形延伸

拉勾教育
—互联网人实战大学—

- 题目变形
  - 本题考点之一就是大数加法问题
  - （练习）两数相减如何实现（进位变借位）？
- 延伸扩展
  - 在面对一些数字类处理的题目时，不妨思考下是否可以利用一些数学的特性来解决问题！
- 本题来源
  - Leetcode 2 https://leetcode-cn.com/problems/add-two-numbers/

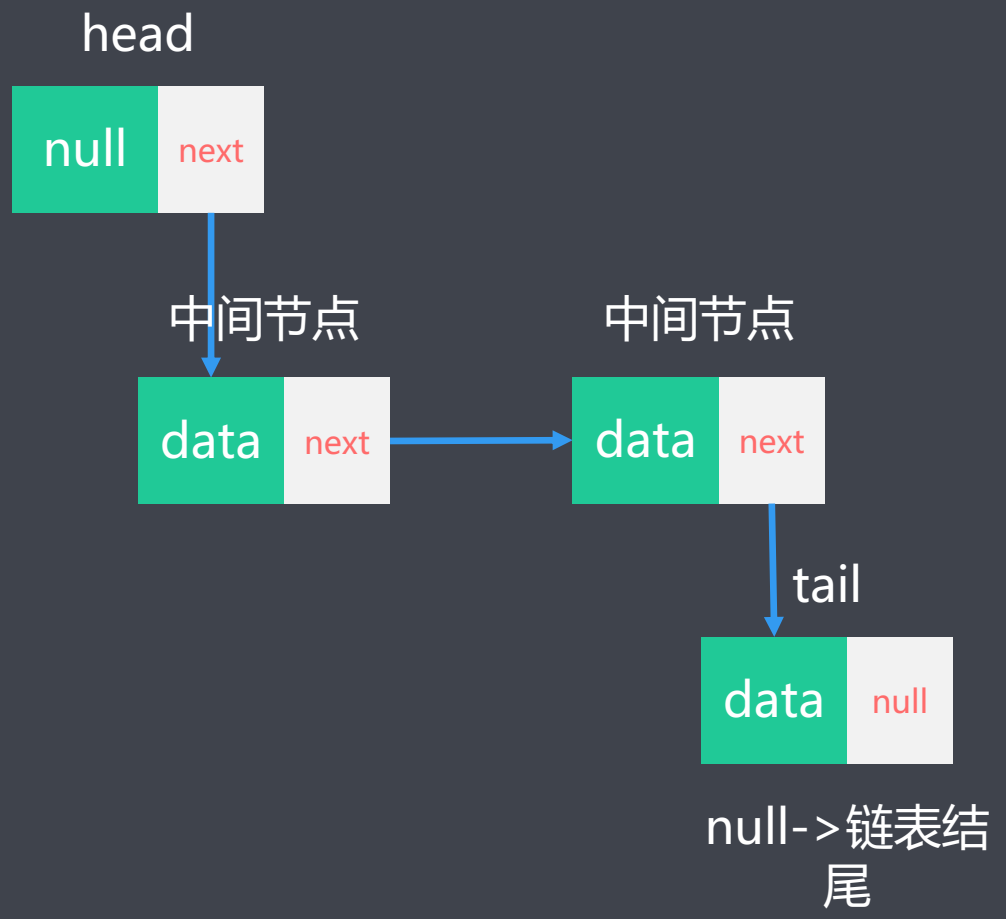
# 总结

拉勾教育
—互联网人实战大学—

## 6C解题法

## 链表的特点


- 数据存储不要求连续空间，不限制容量
- 数据的逻辑顺序通过指针链接次序实现
- 从链表头部依次访问后面的节点
- 在链表表头插入数据的时间复杂度是O(1)




## 数学思维的应用

- 数学竖式

# 课后练习



1. 返回两个二进制字符串的和（用二进制表示）(Leetcode 67/简单)
2. 不使用运算符 + 和 - ，计算两整数 a 、b 之和（Leetcode 371/简单）
3. 两数相加II（Leetcode 445 /中等）
4. 数组形式的整数加法（Leetcode 989/简单）

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容