首先，在这个算法解决过程中，我学会了单向链表的实现。

1.链表节点的结构体：

public static class ListNode{ //构造链表节点结构体  
 int val;  
 ListNode next;  
 public ListNode(){}  
 public ListNode(int val){  
 this.val=val;  
 }  
}

注意构造函数一定要加入无参数构造，第一次我写的时候没有写默认的无参数构造就造成了一些困惑和问题，虽然默认无参数构造会默认给出，但是给出的条件是你没有写别的构造函数，我写了有参数的构造函数，系统默认就不给出无参数的构造函数了，所以需要自己写无参数的构造函数。写构造函数的时候应该一步一步认真的思考后在写。

2.链表的构造：

public static ListNode listBuilder(int[]nums){

//链表节点生成链表  
 ListNode head , last , newnode; //初始化  
 head = new ListNode();  
 head.next=null;  
 last = new ListNode();  
  
 for(int i=0;i<nums.length;i++){  
 newnode = new ListNode(nums[i]);  
 if(head.next==null){  
 head.next=newnode;  
 last=newnode;  
 last.next=null;  
 }  
 else{  
 last.next=newnode;  
 newnode.next=null;  
 last=newnode;  
 }  
 }  
 return head;  
}

在链表构造过程中，开始我所用的节点忘记初始化了，导致程序虽没有错误但是不能运行。在链表的构造过程中，先做连接链表，在把last和要连接链表进行替换。注意，尾巴一定要（last.next）赋值为null。

3.遍历链表

public static void showList(ListNode ln){  
 ListNode p = new ListNode();  
 p = ln.next;  
 while(p!=null){  
 System.*out*.print(p.val+" ");  
 p = p.next;  
 }  
 System.*out*.println();  
}

每个链表都有头结点的，但是头结点只是头结点用于指针的指引作用，并没有数据存储。遍历链表的话需要跳过头结点。

4.添加链表节点

public static void addNode

(int index,intval,ListNode head){ ListNode addnode = new ListNode(val);

//创建要加入的数值的节点  
 int i = 0 ;  
 ListNode p = new ListNode();  
 p = head;  
 while (p!=null){  
 if(i==index-1){  
 addnode.next = p.next;  
 p.next = addnode;  
 }  
 i=i+1;  
 p = p.next;  
 }  
}

添加过程中，要先连链再断开，万万不可先断再连，先断的话原来的链子就找不到了。添加节点其实分两种情况：空链、非空链。非空时候注意头结点虽然没有数据但是遍历的时候不注意可能被记录进去，我这种方法先遍历后计数，count从头结点开始计数必须从0开始，若从头节点的下一个节点开始，就是有数据的节点，必须要考虑情况是在第一个数据节点前插入节点。推荐直接从头结点开始计数，计数以0开始。

4.删除节点

public static void removeNode(int index,ListNode head){  
 int i = 0;  
 ListNode p = new ListNode();  
 p = head;  
  
 while(p!=null){  
 if(i==index-1){  
 p.next=p.next.next;  
 }  
 i=i+1;  
 p=p.next;  
 }  
}

讨论完单链表，我们回到算法题中来

题目：

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.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example**

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

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

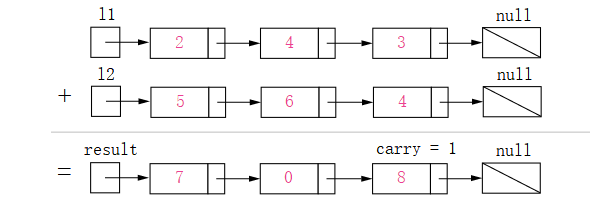
**Explanation:** 342 + 465 = 807.

我的想法：

只是把链表内容提取出来，然后倒置计算成相应的整数，再相加再倒置。AddTwoNumbers是我写的，但是并没有运行出来。很麻烦

答案：

思路：



如同加法演算一样，把每个位置的数字相加，然后进位，并把最终结果输入到一个链表中。

所以有三个变量，x为L1的数值，y为L2的数值，还有有进位的数值carry

sum=x+y+carry。

每个相邻位进制为10，向下个进位的carry值就是sum / 10 。 / 除法取整

本身保留的数值为 sum % 10 。 % 除法取余

但是同时判断两个链表的长度，以比较长的为准，短的不够长度的节点数值用0代替。

遍历下一个节点的时候如果到头了p.next=null的时候不继续往下遍历了，就是以后的数值直接取0就可以了。

源代码：

public static ListNode addTwoNumbers(ListNode l1, ListNode l2){  
  
 int sum=0;  
 int carry=0; //进位  
  
 ListNode p1 = new ListNode();  
 ListNode p2 = new ListNode();  
 p1 = l1.next;  
 p2 = l2.next;  
  
 ListNode head = new ListNode();  
 head.next=null;  
 ListNode last = new ListNode();  
 ListNode newnode;  
  
 while(p1!=null || p2!=null){  
  
 int x = (p1 != null) ? p1.val : 0;  
 int y = (p2 != null) ? p2.val : 0;  
 sum = carry + x + y;  
 carry = sum / 10 ;  
  
 newnode = new ListNode(sum % 10);  
 if(head.next==null){  
 head.next=newnode;  
 last=newnode;  
 last.next=null;  
 }  
 else {  
 last.next=newnode;  
 newnode.next=null;  
 last=newnode;  
 }  
 if(p1 != null)  
 p1 = p1.next;  
 if(p2 != null)  
 p2 = p2.next;  
 }  
  
 return head;  
}