**题目描述**

输入一个链表，从尾到头打印链表每个节点的值。

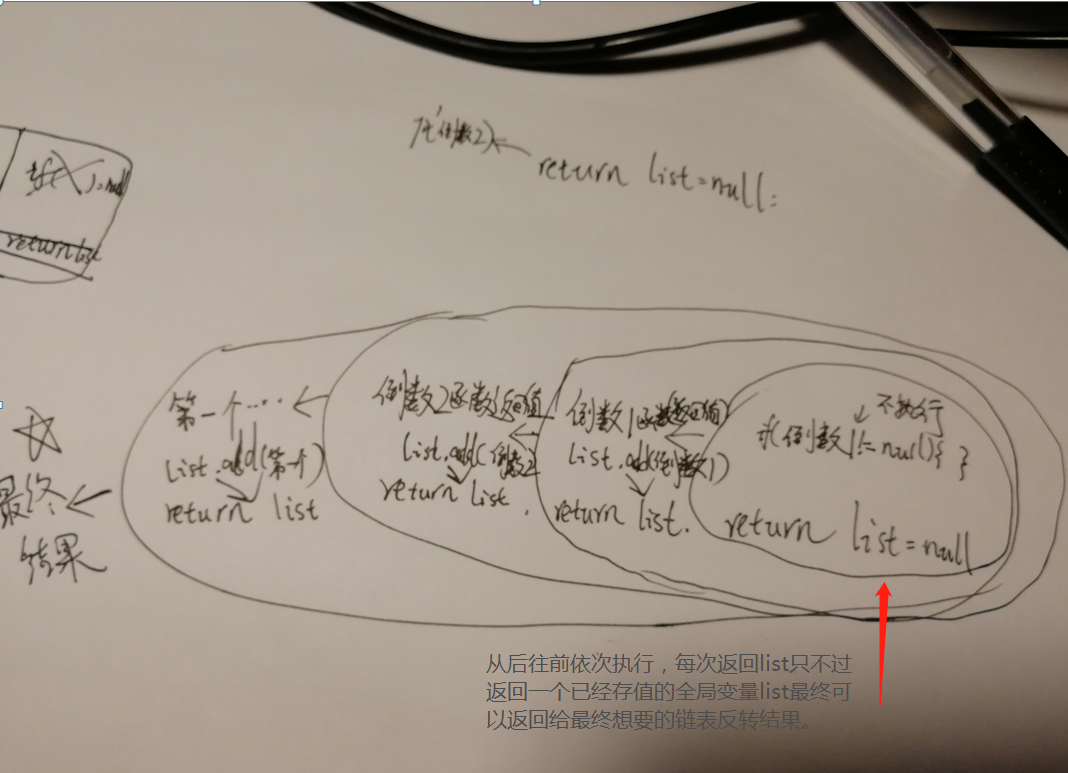
除了可以用翻转链表外，还有两种方法，递归与栈。

两种思路：一种递归。一种利用栈先进后出的特点，先存再取。我用后者先写出来。

重点讲一下递归的（渣渣思考了挺久的）：

**package** facehandjava.Linked;  
  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.Iterator;  
**import** java.util.List;  
  
**public class** DGReversal {  
 **public static void** main(String[] args) {  
  
 Node n6 = **new** Node(11,**null**);  
 Node n5 = **new** Node(9, n6);  
 Node n4 = **new** Node(7, n5);  
 Node n3 = **new** Node(5, n4);  
 Node n2 = **new** Node(3, n3);  
 Node n1 = **new** Node(1, n2);  
*// n6.setNext(n4);* System.***out***.println(**"++++++++++++前++++++++++++"**);  
 Node now = n1;  
 **while** (now != **null**) {  
 System.***out***.print(now.getVal()+**","**);  
 now = now.getNext();  
 }  
 *list* = *DGReversal*(n1);  
 System.***out***.println();  
 System.***out***.println(**"++++++++++++后++++++++++++"**);  
 **for** (Integer integer : *list*) {  
 System.***out***.print(integer+**","**);  
 }  
 }  
  
 **public static** ArrayList<Integer> *list* = **new** ArrayList<>();  
 **public static** ArrayList<Integer> DGReversal(Node node) {  
 **if** (node != **null**) {  
 *DGReversal*(node.getNext());  
 *list*.add(node.getVal());  
 }  
 **return** *list*;  
 }  
}

大概解释一下，就是链表节点不为空的话，节点指向next继续循环那个函数，直到循环完最后一个节点，最后节点.next为空了，函数开始返回list=null给最后节点的函数，意味着最后一个节点的函数结束，并且执行list.add（最后节点的值），然后返回这个list给倒数第二节点函数；倒数第二个节点收到最新return的list后也意味着它的函数结束，再执行list.add……重复这个过程直到第一个节点函数结束，再list.add第一个节点，最终返回list。得到链表的倒置。估计很难看文字看懂，直接上图吧，手稿，有点大鱼吃小鱼的感觉。



/\*\*

\*       栈比较直观也比较简单，就直接摆上代码了。

\*   public class ListNode {

\*       int val;

\*       ListNode next = null;

\*

\*       ListNode(int val) {

\*           this.val = val;

\*       }

\*   }

\*

\*/

import java.util.ArrayList;

import java.util.Stack;

public class Solution {

   public ArrayList<Integer> printListFromTailToHead(ListNodelistNode) {

       ArrayList<Integer> list = new ArrayList<>();

       Stack<Integer> stack = new Stack<Integer>();

       while(listNode!=null){

           stack.push(listNode.val);

           listNode=listNode.next;

       }

       while(!stack.isEmpty()){

           list.add(stack.pop());

       }

       return list;

    }

}