# 题目

给您一个不可变的链表，使用下列接口逆序打印每个节点的值：

ImmutableListNode: 描述不可变链表的接口，链表的头节点已给出。

您需要使用以下函数来访问此链表（您 不能 直接访问 ImmutableListNode）：

ImmutableListNode.printValue()：打印当前节点的值。

ImmutableListNode.getNext()：返回下一个节点。

输入只用来内部初始化链表。您不可以通过修改链表解决问题。也就是说，您只能通过上述 API 来操作链表。

进阶：

您是否可以：

使用常数级空间复杂度解决问题？

使用线性级时间复杂度和低于线性级空间复杂度解决问题？

示例 1：

输入：head = [1,2,3,4]

输出：[4,3,2,1]

示例 2：

输入：head = [0,-4,-1,3,-5]

输出：[-5,3,-1,-4,0]

示例 3：

输入：head = [-2,0,6,4,4,-6]

输出：[-6,4,4,6,0,-2]

提示：

链表的长度在 [1, 1000] 之间。

每个节点的值在 [-1000, 1000] 之间。

# 分析

## 方法一：递归

/\*\*

\* // This is the ImmutableListNode's API interface.

\* // You should not implement it, or speculate about its implementation.

\* class ImmutableListNode {

\* public:

\* void printValue(); // print the value of the node.

\* ImmutableListNode\* getNext(); // return the next node.

\* };

\*/

class Solution {

public:

void printLinkedListInReverse(ImmutableListNode\* head) {

if (head == NULL) { return; }

printLinkedListInReverse(head->getNext());

head->printValue();

}

};

## 方法二：双指针

这个题解其实利用了“找出倒数第n个节点”这道题的思路，每次让n加一，直到n等于链表长度为止。

我们引入 fast\_begin 变量，开始时指向 head 的下一个节点，作为每次迭代时快指针 fast 开始的位置，慢指针 slow 每次都指向 head 节点。让 fast 和 slow 每次各走一步，当 fast 走到链表尽头时，慢指针 slow 指向的节点就是我们要找的节点。打印 slow 之后，即可让 fast\_begin 前进一步，然后重复以上步骤，直到 fast\_begin 也走到尽头。

不要把 head 给漏掉了，打印之即可。

代码如下：

class Solution {

public:

void printLinkedListInReverse(ImmutableListNode\* head) {

if (head == nullptr)

return;

ImmutableListNode\* fast\_begin = head->getNext();

while (fast\_begin != nullptr) {

ImmutableListNode\* slow = head;

ImmutableListNode\* fast = fast\_begin;

while (fast != nullptr) {

slow = slow->getNext();

fast = fast->getNext();

}

slow->printValue();

fast\_begin = fast\_begin->getNext();

}

head->printValue();

}

};

这个方法只额外引入了三个指针（引用）变量，未使用额外的数据结构，也未使用递归，所以空间复杂度为O(1)，但是每打印完一个节点，都需要重新遍历一遍链表的剩余部分，所以时间复杂度为 O(n^2)，典型的时间换空间思路。