栈

栈（Stack）是一种基本的数据结构，它按照先进后出（Last In, First Out，LIFO）的原则管理数据。栈可以看作是一种限制性的线性表，只允许在表的一端（称为栈顶，Top）进行插入和删除操作。

栈顶和栈底的概念：

栈顶（Top）： 栈中最新添加的元素所在的位置。新元素入栈时，它成为新的栈顶。

栈底： 栈中最早添加的元素所在的位置。栈底的元素是最后一个被移除的元素。

(为了避免中文发音 “顶” 和 “底” 可能存在的混淆，以后只用stack top, stack bottom来说明避免中文发音)

栈的基本操作有两个：

入栈（Push）： 将元素添加到栈顶。

出栈（Pop）： 从栈顶移除元素。

定义单调栈的单调，指从stack top -> stack bottom的顺序。

此后，在js中我们想象栈的操作时候，脑袋里统一把stackTop放在左边，把stackBottom放在右边。

[stackTop, ..., ..., stackBottom]

但是在代码层面的操作上，不要用 unshift + shift。 要使用push + pop。

(

在数组开头进行 unshift 和 shift 操作的时间复杂度是 O(n)，因为这些操作会导致整个数组的重新索引。

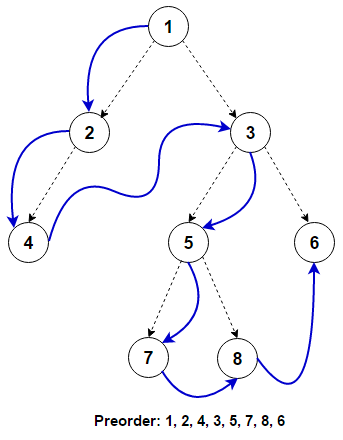
在数组末尾进行 push 和 pop 操作的时间复杂度是 O(1)，因为它们不涉及整个数组的移动。

)

树

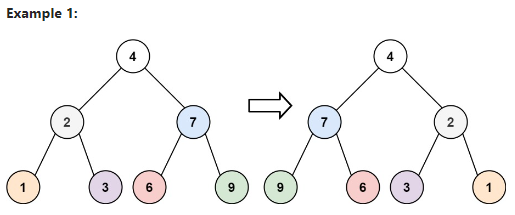
BinaryTreePreorderTraversal

二叉树前序遍历 根-左-右 (DLR)



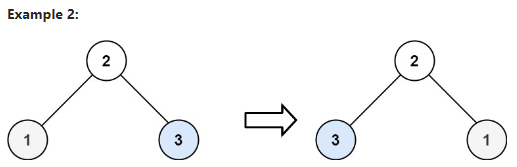
226. Invert Binary Tree

翻转二叉树



Input: root = [4,2,7,1,3,6,9]

Output: [4,7,2,9,6,3,1]



Input: root = [2,1,3]

Output: [2,3,1]

链表

[leetcode] 0234回文链表 js实现 图解

/\*

给你一个单链表的头节点 head ，请你判断该链表是否为回文链表。如果是，返回 true ；否则，返回 false 。

\*/

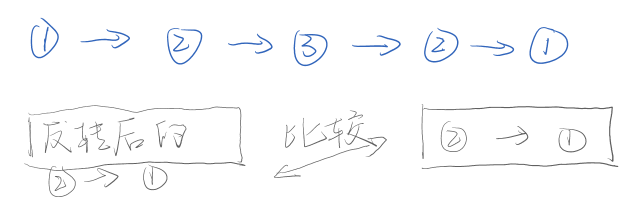
前置知识： lc0876 链表的中间节点 (用快慢指针找到链表的中间节点)

Lc0234这题的的思路是

1、用快慢指针找到原链表后半部分(找到后半段链表的头指针)。

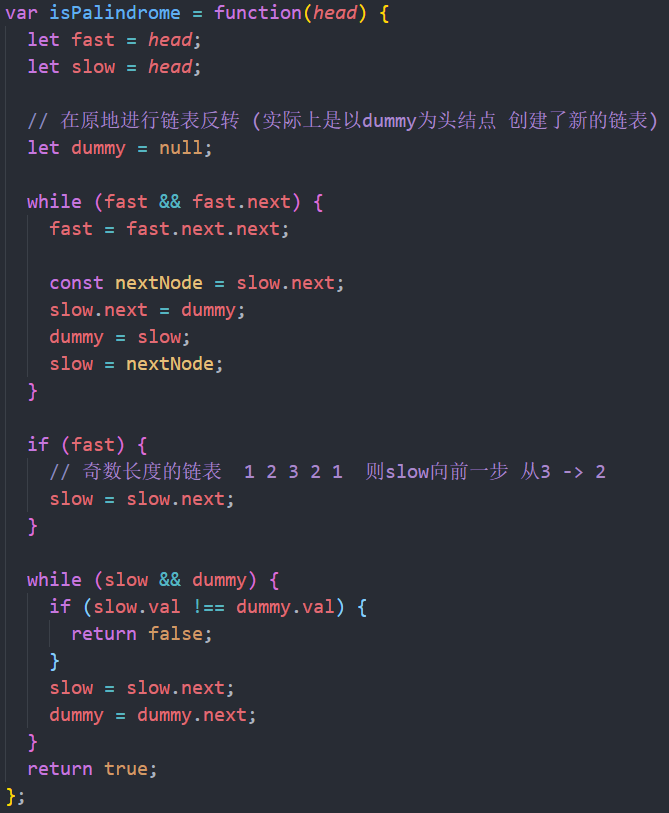
2、将前半段链表反转。

3、比较反转后的前半段链表和后半段链表。



其中1、2 两步骤可以同时进行。

先看代码

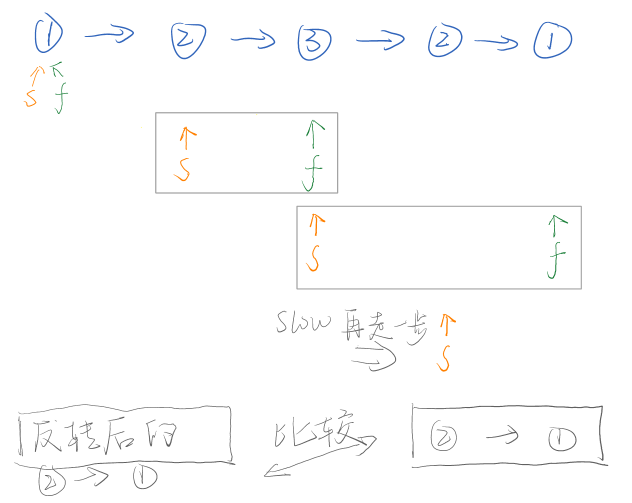


这题有两个需要注意的点

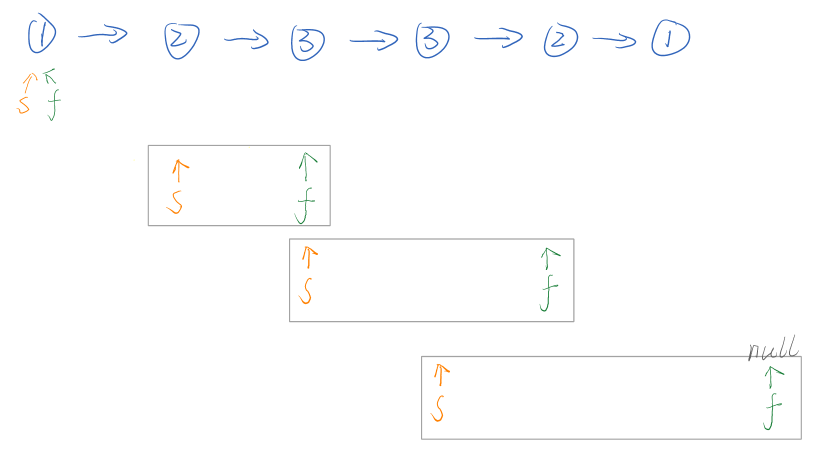
1. 奇数长度的链表和偶数长度的链表处理方式不同。
2. 反转链表的操作本身。

下面看图解。

奇数长度的链表 slow需要往前走一步

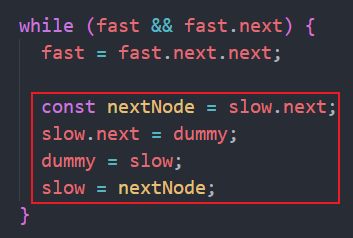


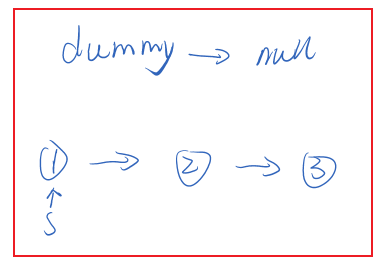
而偶数长度的链表，快慢指针的while循环之后slow指针已经到达了预期的位置

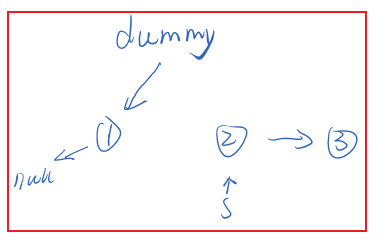


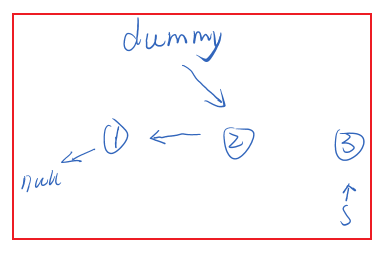
再看反转链表的过程，这是刚接触链表的选手比较难理解的地方

重点在while循环中红框的代码部分









[leetcode] 0092反转链表2 js实现 图解

/\*

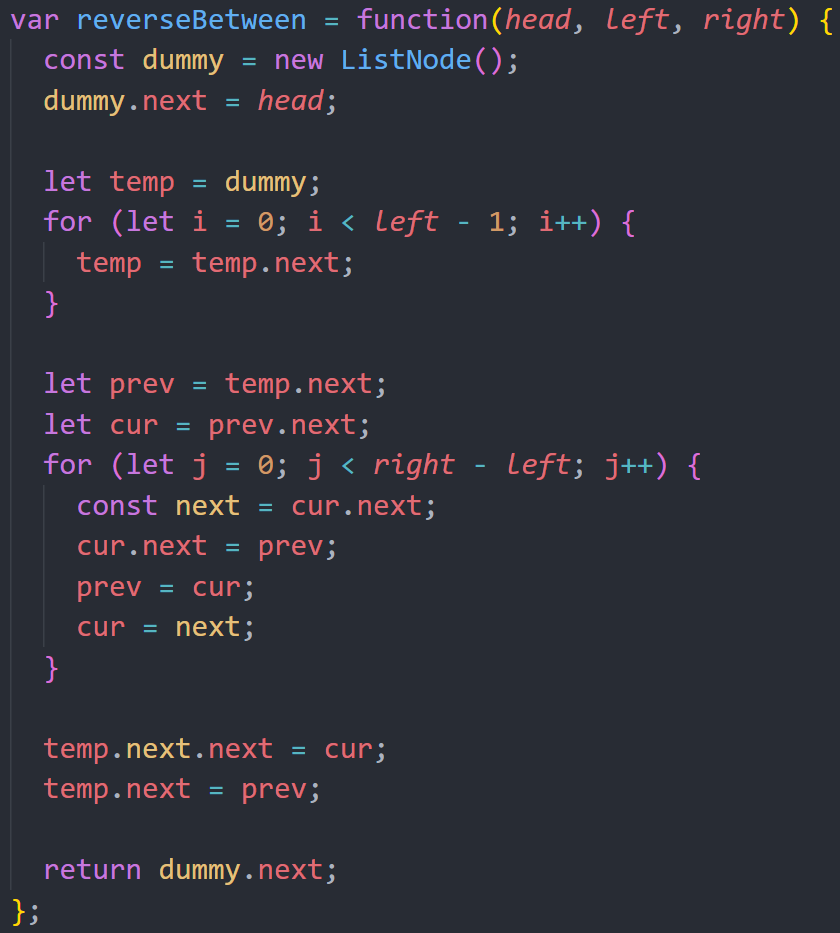
给你单链表的头指针 head 和两个整数 left 和 right ，其中 left <= right 。请你反转从位置 left 到位置 right 的链表节点，返回 反转后的链表 。

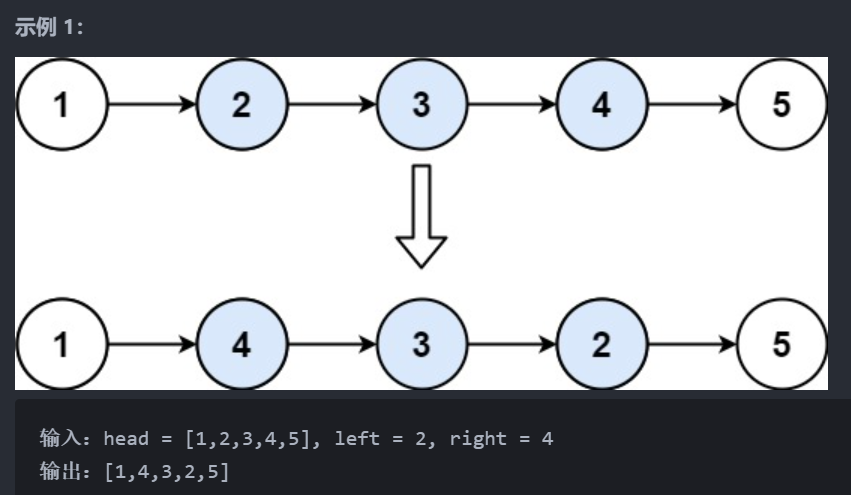
\*/

前置知识： lc0206 反转链表

Lc0092这题的的思路和206题的反转链表一样，难在操作。

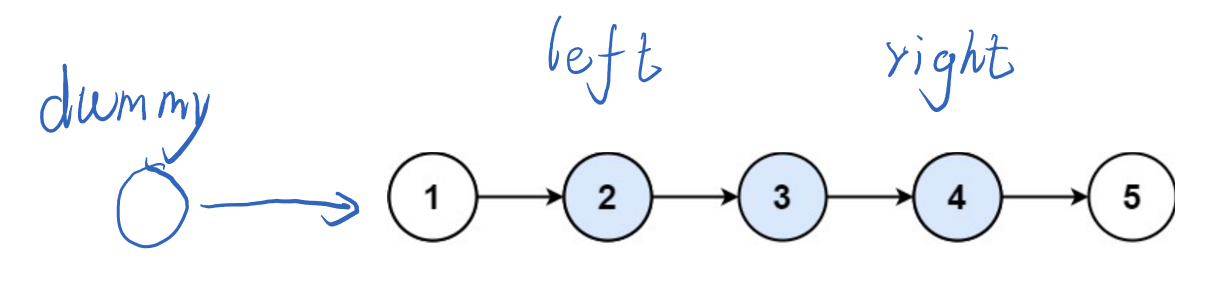
先贴代码



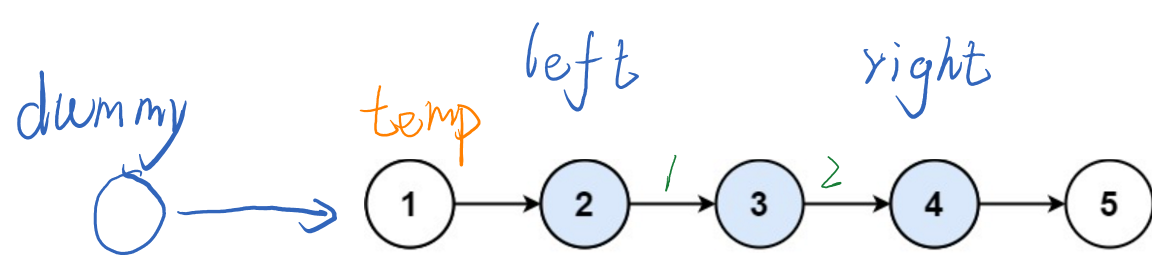


拿上示例，来看图解

1、首先是拼接上dummy，最终返回dummy.next。(这是常规操作)

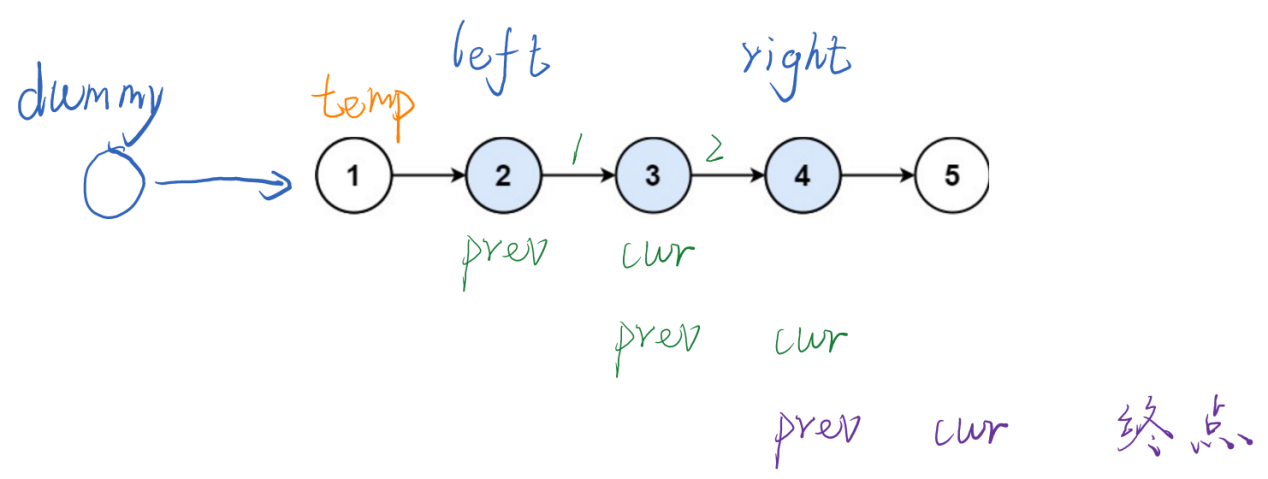


1. 第一个for循环，让temp从dummy位置出发，移动到left位置的上一个节点



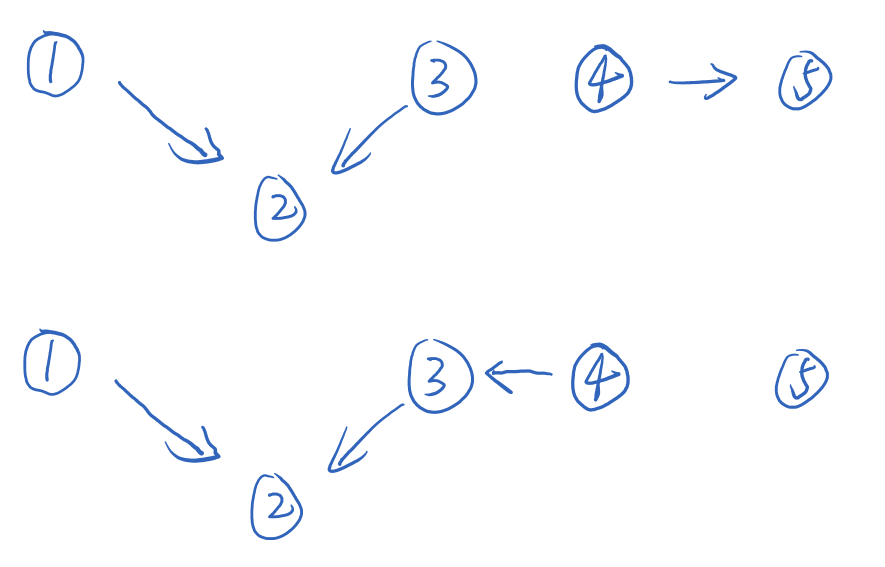
注意，此后temp的位置没有移动，temp很长一段时间内指向也没有变(1 -> 2)。

1. 第二个for循环，操作prev和cur，一边移动位置，一边反转节点指向。

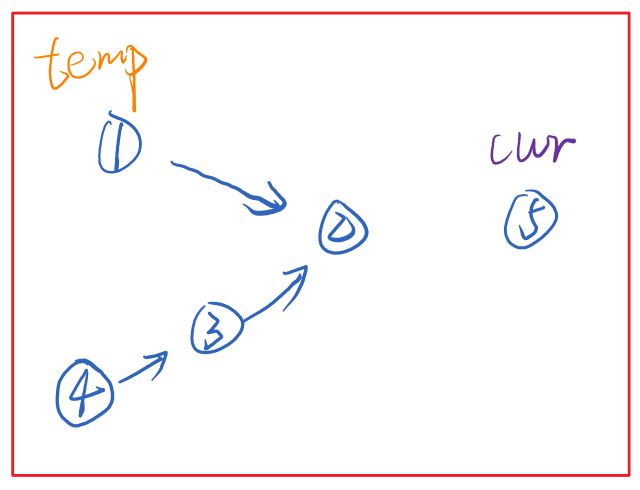


其中指向改变了(right - left)次，第二个for循环终点时prev处于right位置，cur处于right.next位置。

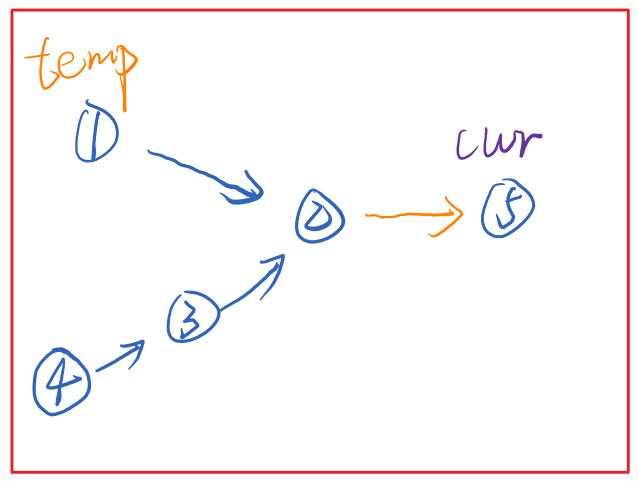
第二个for循环完成之后链表中各个节点的指向是下面这样的



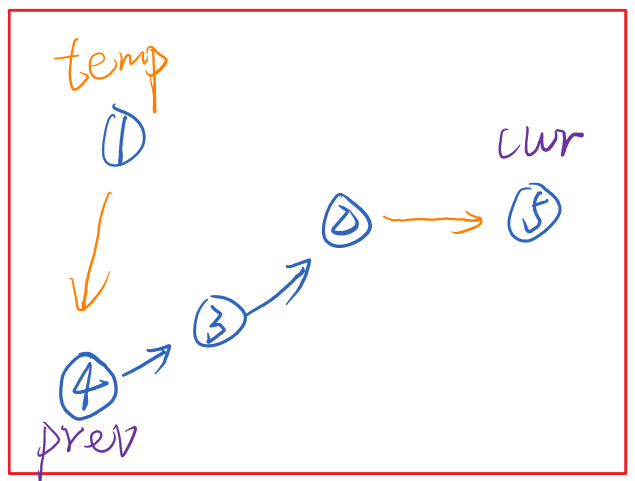
1. 重新画一下图，看一下现在的链表指向



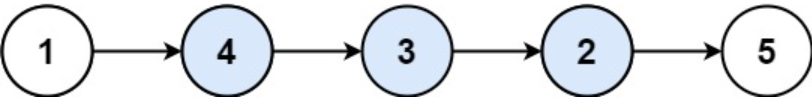
1. temp.next.next = cur



1. temp.next = prev



已经反转完成了



返回dummy.next，完结。