极客学院
jikexueyuan.com

链表环与链表交点

链表环与链表交点 — **课程概要**

- 约瑟夫环
- 链表交点
- 判断链表是否有环
- 链表环的起始节点
- 寻找重复元素

约瑟夫环

约瑟夫环

- 问题描述
- 思路分析
- 创建循环链表
- 代码实现
- 测试用例
- 其它解法

约瑟夫环 — 问题描述

丢手绢：

约瑟夫环 — 问题描述

约瑟夫环问题：

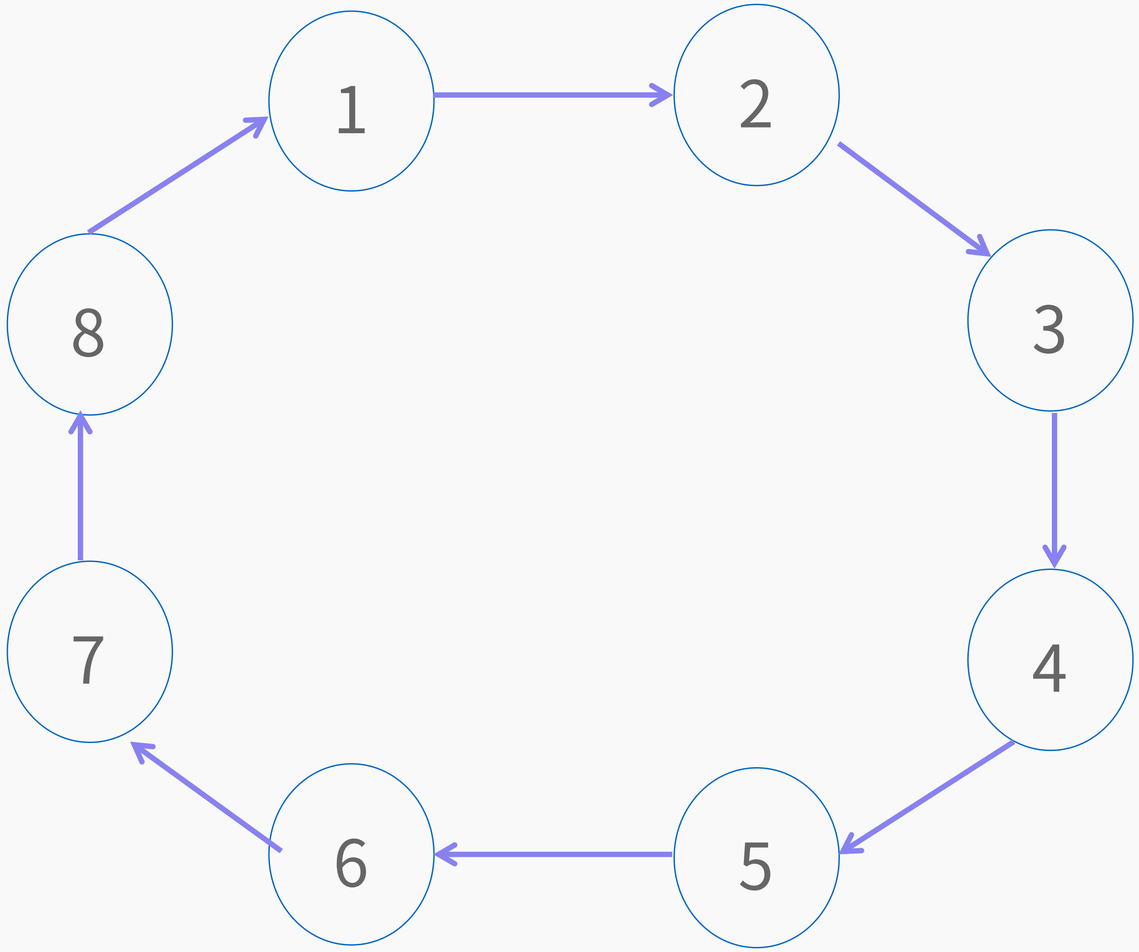
- n个人围在一起形成圆环
- 从某个编号（start）开始报数
- 数到某个数（step）的时候，此人出列，下一个人重新从1开始报数
- 循环执行第三步，直到所有人都出列，游戏结束

要求：编写程序，打印出列顺序

编号、报数等等都从1开始

约瑟夫环 — 问题描述

n = 8，start = 3，step = 4



6　2　7　4　3　5　1　8

约瑟夫环 — **思路分析**

用**循环链表**模拟报数、出队过程：

- 创建循环链表
- 找到编号为start的节点
- 打印并删除第step-1个节点
- 从下一个节点重新开始计数，循环执行第三步，直到链表为空

约瑟夫环 — 创建循环链表

如何创建约瑟夫环?

Key	Value
类名	ListNode
方法名	arrayToCircle
作用	根据数组，创建循环链表

约瑟夫环 — 代码实现

Key	Value
类名	Josephus
方法名	josephusCircle
时间复杂度	O(N*Step)
空间复杂度	O(N)

约瑟夫环 — 测试用例

Key	Value
类名	Josephus
方法名	test
测试输入	{1,2,3,4,5,6,7,8}
测试输出	6,2,7,4,3,5,1,8

约瑟夫环 — **其它解法**

只需要求**最后一个出队的人**，更快的方法，时间复杂度为O(N)，空间复杂度为O(1)。

这种方法的重点在于**推导**，不在于**编程**。

面试官注重考查应聘者对循环链表的掌握。

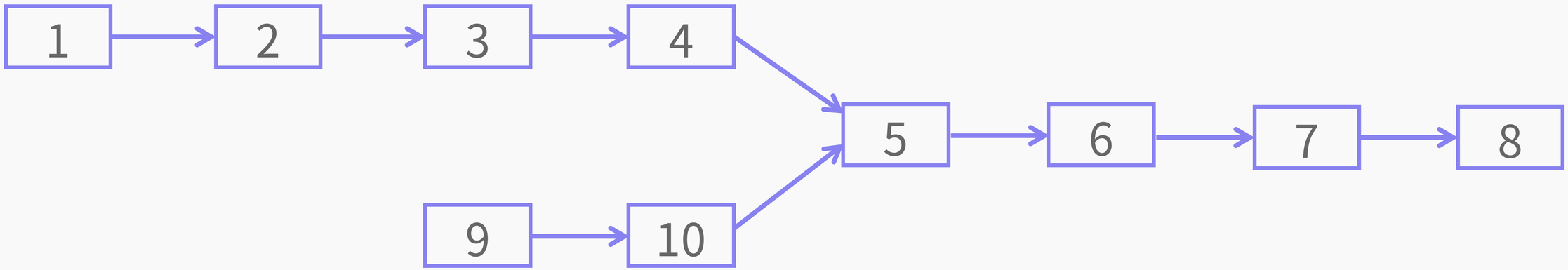
链表交点

链表交点

- 问题描述
- 创建链表交点
- 暴力求解法
- 利用哈希表
- 线性算法

链表交点 — 问题描述

两个链表，从某个节点开始，后面的节点均相同，则称这个节点为两个链表的交点。



链表交点 — 问题描述

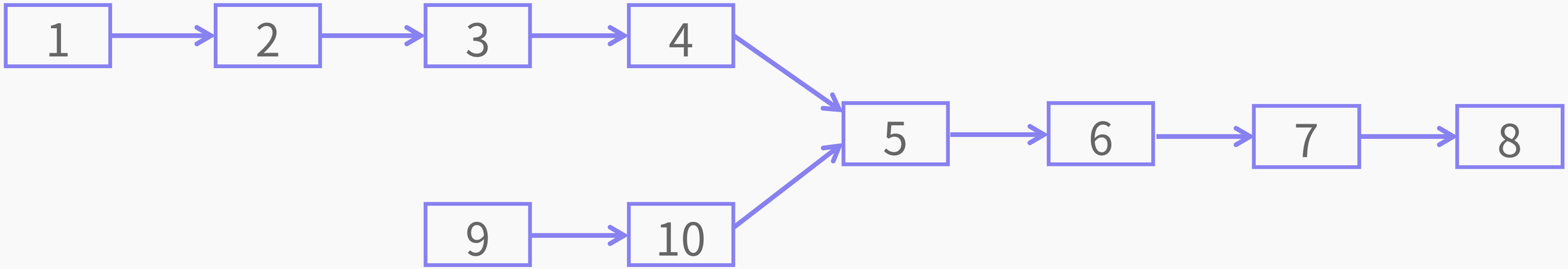
leetCode 160：Intersection of Two Linked Lists

给定两个单链表，寻找它们的交点。

额外要求：

- 如果没有交点，返回null
- 不允许改变链表的结构
- 假设链表没有环
- 时间复杂度尽量为O(N)，空间复杂度尽量为O(1)

链表交点 — 创建链表交点



Key	Value
类名	ListNode
方法名	arrayToIntersection
作用	根据两个数组，创建有交点的两个链表

链表交点 — **暴力求解法**

遍历第一个链表，依次判断遍历到的节点是否能在第二个链表中找到。

```
for(p in list1){
    for(q in list2){
        if(p==q){
            return p;
        }
    }
}
return null;
```

链表交点 — **暴力求解法**

Key	Value
类名	_160IntersectionOfTwoLinkedLists
方法名	bruteForce
时间复杂度	O(M*N)，M和N表示链表的长度
空间复杂度	O(1)

链表交点 — **利用哈希表**

利用哈希表优化时间复杂度。

```
hashSet();
for(p in list2){
    hashSet.add(p);
}
for(p in list1){
    if(hashSet.contains(p)) : return p;
}
return null;
```

链表交点 — **利用哈希表**

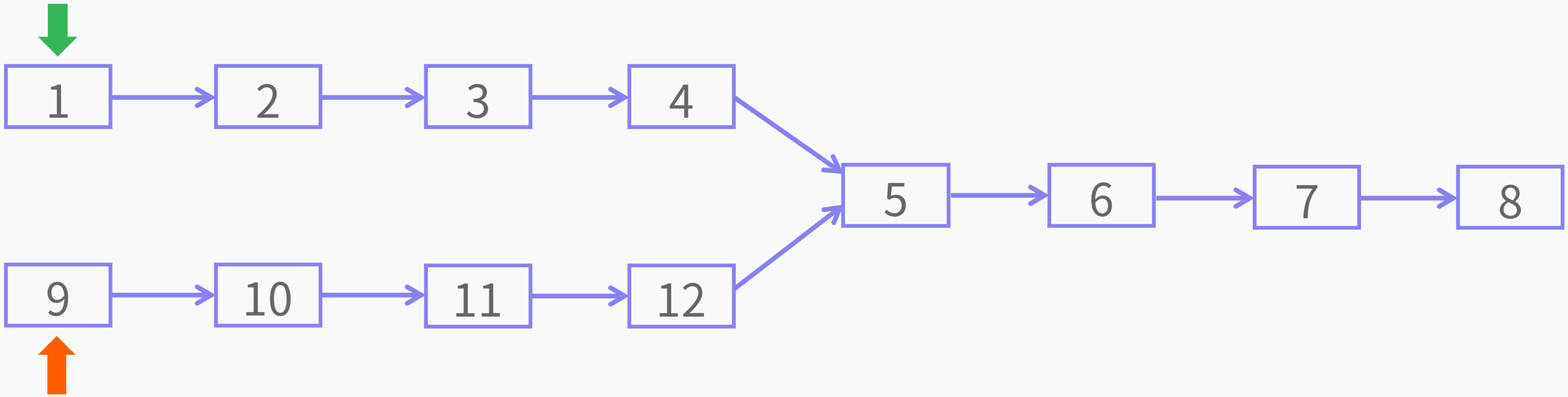
Key	Value
类名	_160IntersectionOfTwoLinkedLists
方法名	hashFunction
时间复杂度	O(M+N)--O(M*sqrt(N))
空间复杂度	O(N)

为什么时间复杂度带有根号？以后再讨论！

链表交点 — **线性算法**

线性复杂度+常数空间

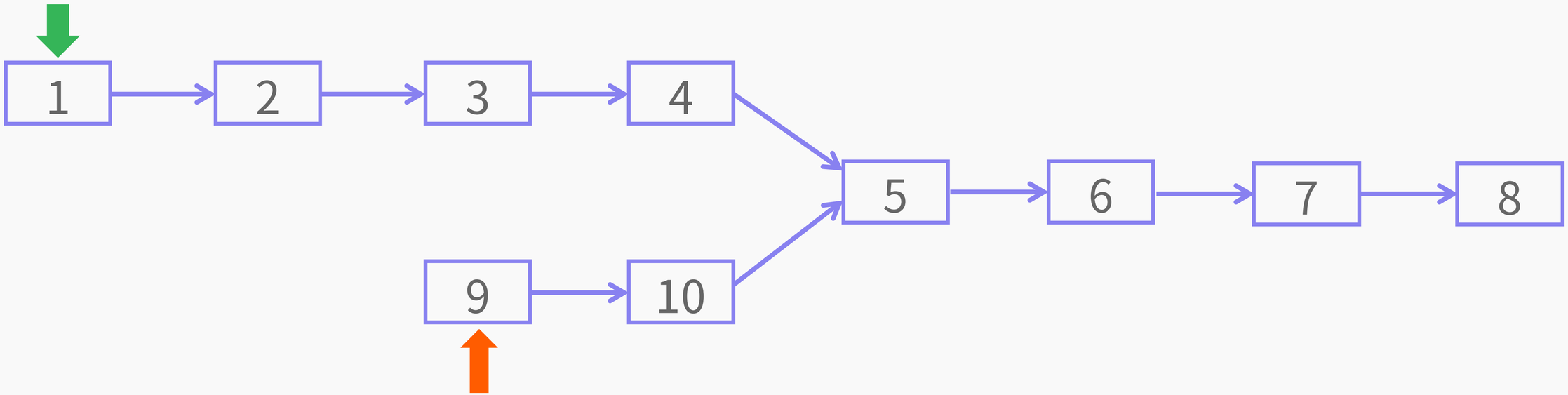
长度相同的两个链表：



链表交点 — **线性算法**

长度不相同的两个链表：

往后走m-n步



链表交点 — **线性算法**

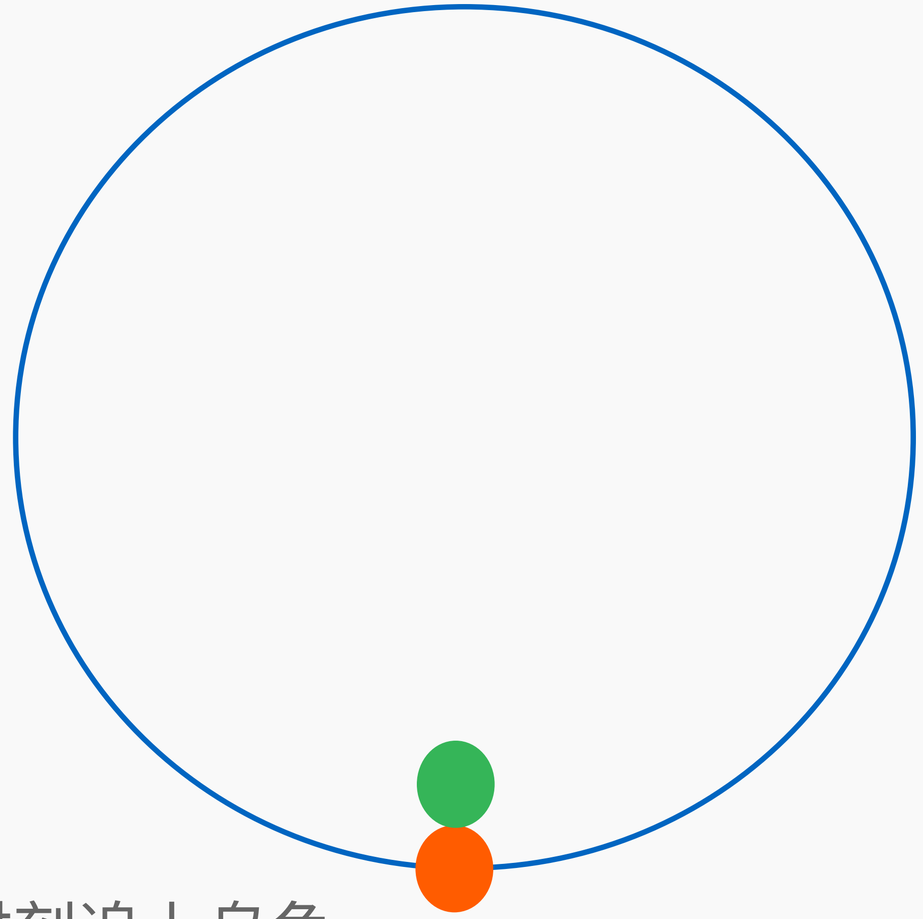
Key	Value
类名	_160IntersectionOfTwoLinkedLists
方法名	getIntersectionNode
时间复杂度	O(M+N)
空间复杂度	O(1)

判断链表是否有环

判断链表是否有环

- 龟兔赛跑原理
- 问题描述
- 思路分析
- 创建链表环
- 代码实现
- 测试与提交

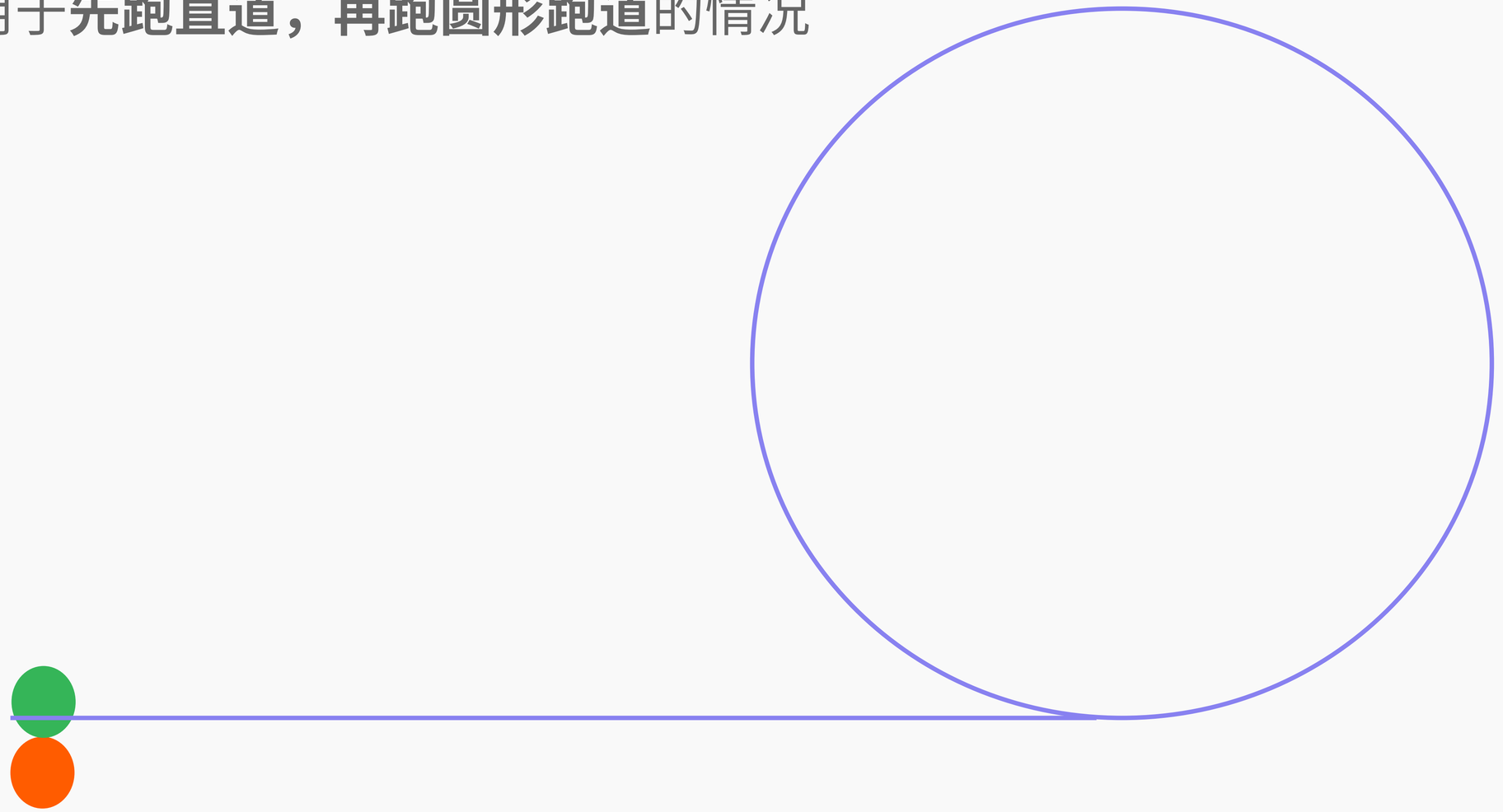
判断链表是否有环 — 龟兔赛跑原理



兔子一定能在某一时刻追上乌龟

判断链表是否有环 — **龟兔赛跑原理**

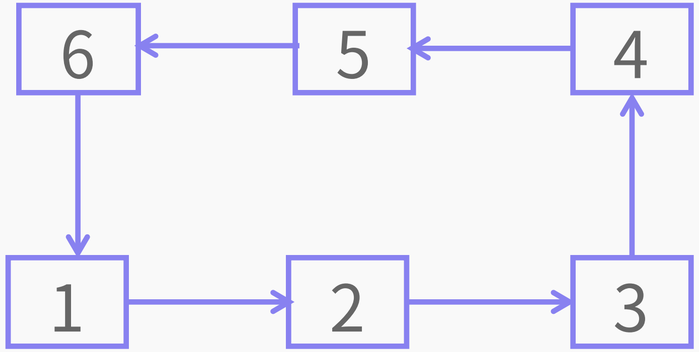
同样适用于**先跑直道，再跑圆形跑道**的情况

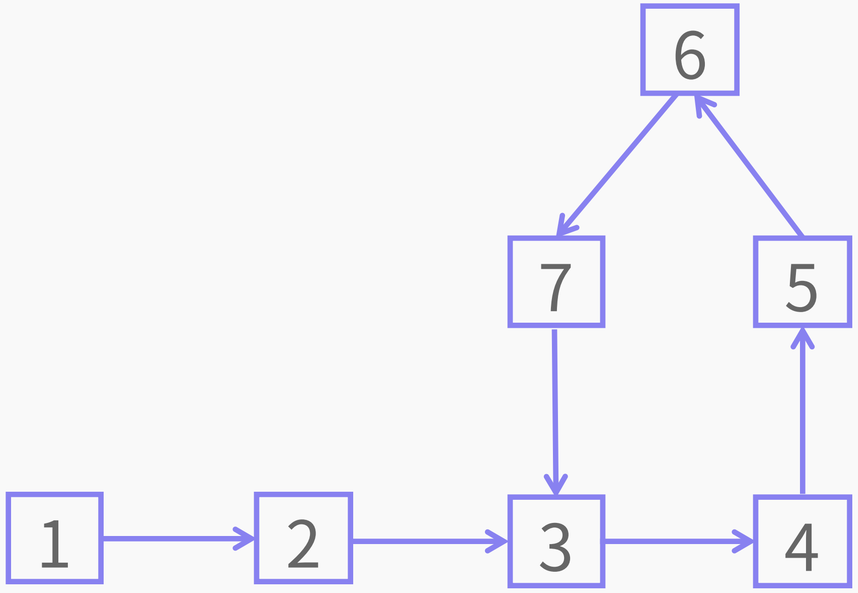


判断链表是否有环 — **问题描述**

leetCode 141：Linked List Cycle

给定单链表，判断它是否有环，要求空间复杂度为O(1)。





判断链表是否有环 — **思路分析**

模拟龟兔赛跑的过程：

- 定义两个指针fast、slow，初始化为head
- 反复执行“fast向后移动两位、slow向后移动一位”，直到fast与slow相遇
- 如果到达链表尾部还未相遇，则不存在环

结束条件：fast不为空且fast的后继也不为空

判断链表是否有环 — **创建链表环**

与“约瑟夫环”的创建类似。

Key	Value
类名	ListNode
方法名	createCycle
作用	根据数组，创建含有环的链表

判断链表是否有环 — **代码实现**

Key	Value
类名	_141LinkedListCycle
方法名	hasCycle
时间复杂度	O(N)
空间复杂度	O(1)

判断链表是否有环 — 测试与提交

Key	Value
类名	_141LinkedListCycle
方法名	test
测试输入1	调用arrayToList()创建普通链表
测试输出1	false
测试输入2	调用arrayToCircle()创建约瑟夫环
测试输出2	true
测试输入3	调用createCycle()创建普通的带环链表
测试输出3	true

链表环的起始节点

链表环的起始节点

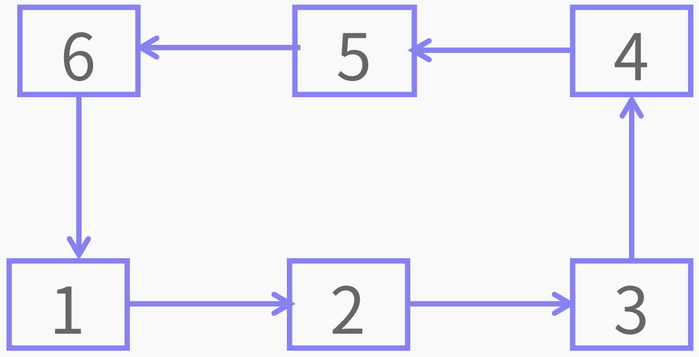
- 问题描述
- 思路分析
- 代码实现
- 测试与提交

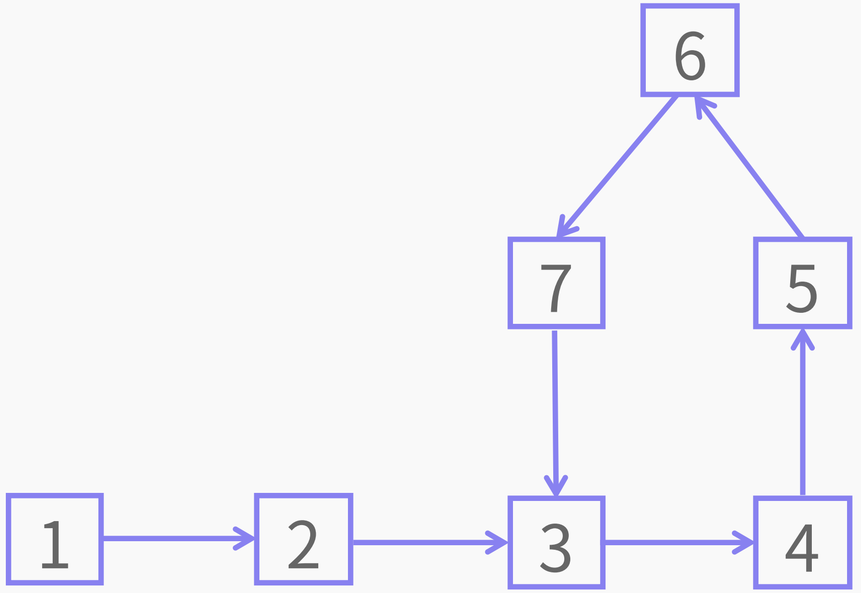
链表环的起始节点 — 问题描述

leetCode 142：Linked List Cycle II

给定单链表，寻找链表环的开始位置；如果没有环，返回空。

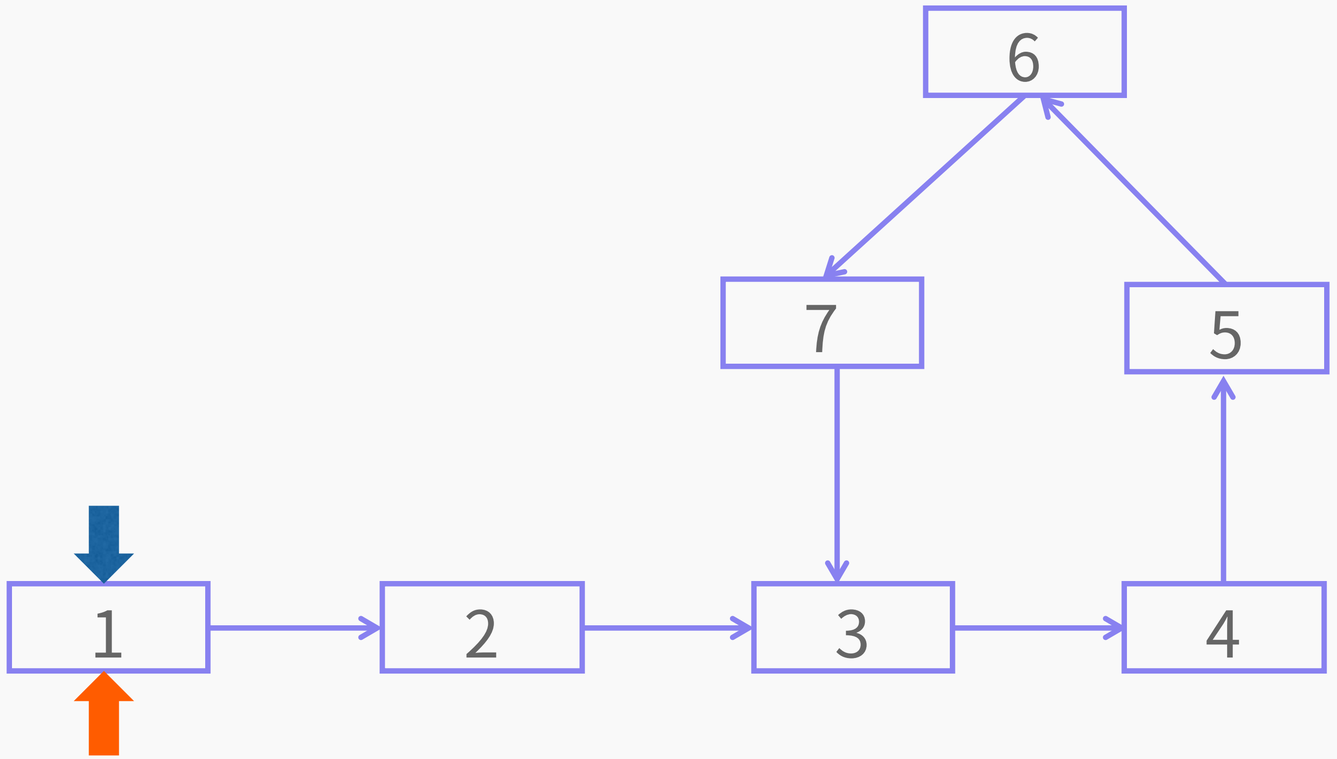
额外要求：空间复杂度为O(1)





链表环的起始节点 — **思路分析**

相遇点到链表环起始节点的距离 = **头结点**到链表环起始节点的距离



链表环的起始节点 — 代码实现

Key	Value
类名	_142LinkedListCycle02
方法名	detectCycle
时间复杂度	O(N)
空间复杂度	O(1)

链表环的起始节点 — 测试与提交

Key	Value
类名	_142LinkedListCycle02
方法名	test
测试输入1	调用arrayToList()创建普通链表
测试输出1	null
测试输入2	调用arrayToCircle()创建约瑟夫环
测试输出2	1
测试输入3	调用createCycle()创建普通的带环链表
测试输出3	3

寻找重复元素

寻找重复元素

- 问题描述
- 暴力求解与哈希表法
- 数组环的思路
- 代码实现
- 测试与提交

寻找重复元素 — 问题描述

leetCode 287：Find the Duplicate Number

给定长度为n+1的整形数组，数组元素的范围在1到n之间，有一个数字重复出现了多次，找出这个数字。

额外要求：

- 数组是只读的，不允许改变数组元素
- 时间复杂度与空间复杂度尽量低
- 只有一个重复的数字，但有可能出现多次

寻找重复元素 — 暴力求解与哈希表法

暴力求解法

时间复杂度O(N^2)

空间复杂度O(1)

```
for(i=0;i<nums.length-1;i++){
    for(j=i+1;j<nums.length;j++){
        if(nums[i]==nums[j]){
            return nums[i];
        }
    }
}
return 0;
```

寻找重复元素 — **暴力求解与哈希表法**

哈希表法

时间复杂度O(N)--O(N*sqrt(N))

空间复杂度O(N)

```
hashSet();
for(i in mums){
    if(!hashSet.contains(i)){
        hashSet.add(i);
    }else{
        return i;
    }
}
return 0;
```

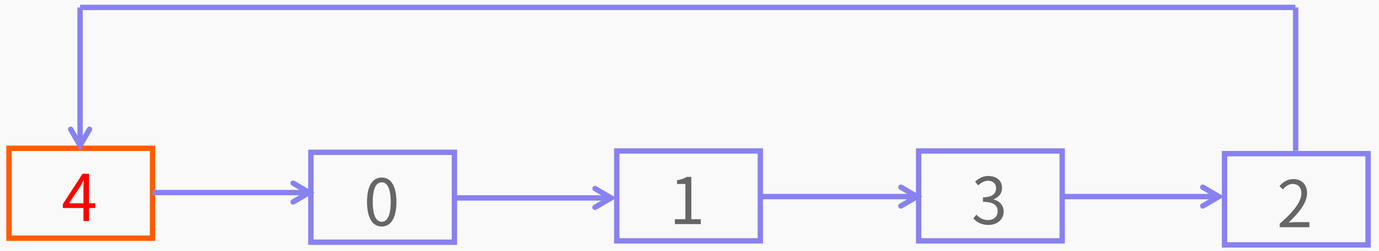
寻找重复元素 — **数组环的思路**

给定乱序、不含有重复数字的数组。

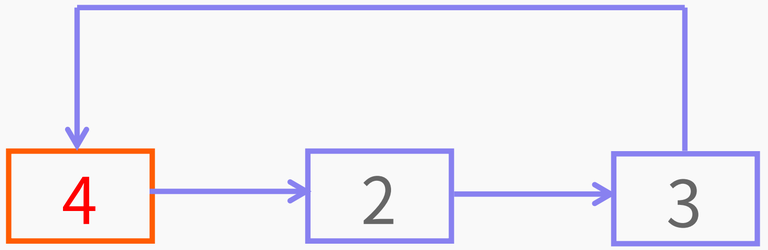
从数组末尾开始，不断的执行**把当前元素值作为下一个下标**的操作。

形成**数组环**。

下标	0	1	2	3	4
元素值	1	3	4	2	0



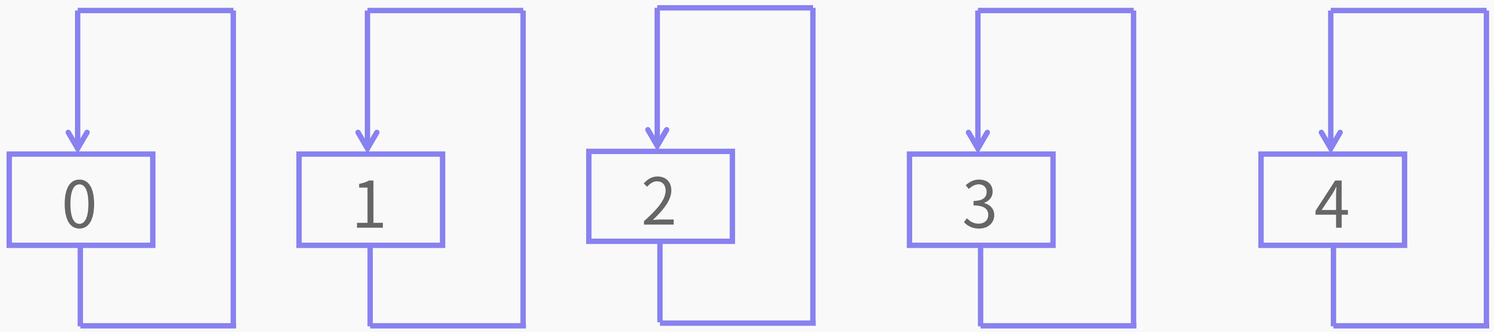
下标	0	1	2	3	4
元素值	1	0	3	4	2



寻找重复元素 — **数组环的思路**

有序数组，任意元素都能形成独立的环。

下标	0	1	2	3	4
元素值	0	1	2	3	4



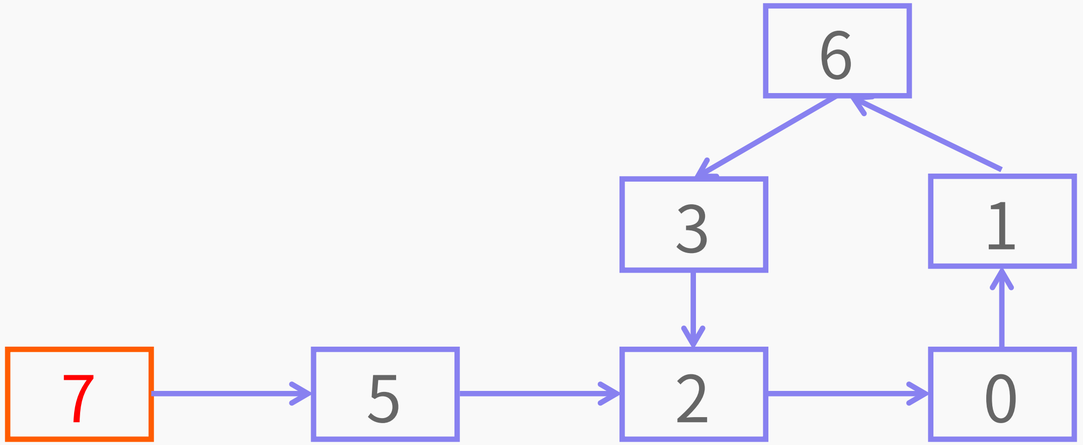
寻找重复元素 — 数组环的思路

给定**含有重复数字**的数组。

从数组末尾开始，不断的执行**把当前元素值作为下一个下标**的操作。

也能形成**数组环**。

下标	0	1	2	3	4	5	6	7
元素值	1	6	0	2	4	2	3	5



寻找重复元素 — **数组环的思路**

求解步骤：

- 假设数组长度为length，初始化快指针fast、慢指针slow为length-1
- 从数组末尾开始，执行index = array[index]，其中fast执行2次
- 不断地执行第2步，直到fast与slow相遇
- 借用**链表环的起始节点**的思路，求出重复元素

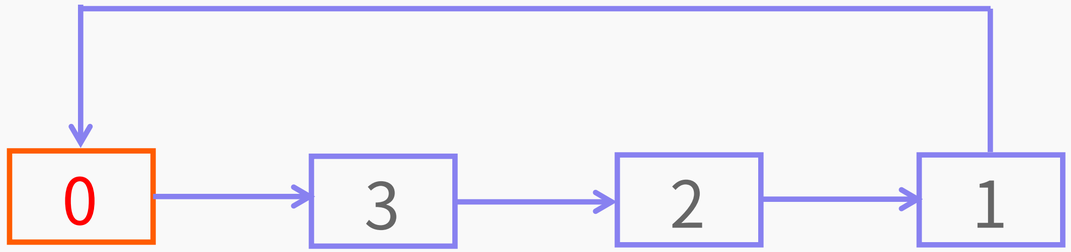
注意：

- 题目给定的元素介于1到n之间
- 不能从数组头部开始

寻找重复元素 — **数组环的思路**

从数组头部开始，形成的数组环有可能**无法包含**重复元素

下标	0	1	2	3	4	5	6	7
元素值	3	0	1	2	6	4	4	5



寻找重复元素 — 代码实现

Key	Value
类名	_287FindTheDuplicateNumber
方法名	findDuplicate
时间复杂度	O(N)
空间复杂度	O(1)

寻找重复元素 — 测试与提交

Key	Value
类名	_287FindTheDuplicateNumber
方法名	test
测试输入1	{2,7,1,3,5,3,4,6}
测试输出1	3
测试输入2	{4,1,2,3,7,5,5,6}
测试输出2	5

链表环与链表交点

本套课程中我们学习了链表环与链表交点。你应当解决了以下面试题：

- 约瑟夫环
- 链表交点
- 判断链表是否有环
- 链表环的起始节点
- 寻找重复元素

你可以使用leetCode验证程序是否正确，还可以在白纸上书写代码；如果想进一步提高，你可以继续在极客学院学习**栈与队列（上）**课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台