141 环形链表

Label：双指针
给定一个链表，判断链表中是否有环。
为了表示给定链表中的环，我们使用整数 pos 来表示链表尾连接到链表中的位置（索引从 0 开始）。 如果 pos 是 -1，则在该链表中没有环。

- 双指针（龟兔赛跑）

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        if (head == null || head.next == null) {
            return false;
        }

        ListNode slow = head;
        ListNode fast = head.next;  // 这里定义fast在下一个节点，不然while循环执行不了，可以假想fast是从head之前的节点条两步过来的，所以也是也一样。当然，也可以用两个do wihle循环，这样两个指针都可以从同一起跑线开始

        while(slow != fast){
            if (fast == null || fast.next == null) {
                return false;
            }
            slow = slow.next;
            fast = fast.next.next;
        }
        return true;
    }
}
```

- Hash==>判断重复，第一想到的肯定是Hash

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        Set<ListNode> nodesSeen = new HashSet<>();
    while (head != null) {
        if (nodesSeen.contains(head)) {
            return true;
        } else {
            nodesSeen.add(head);
        }
        head = head.next;
    }
    return false;

    }
}
```

- 一种会破坏链表结构的方法 （逐步蚕食）

```
public class Solution {
    public boolean hasCycle(ListNode head) {
       while(head != null){
            if(head == head.next){
                return true;
            }
            if(head.next != null){
                head.next = head.next.next;
            }
            head = head.next;
        }
        return false;
    }
}
```

- 双指针2

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        if (head == null || head.next == null) {
            return false;
        }
        ListNode slow = head;
        ListNode fast = head;
        while(fast.next != null && fast.next.next!= null){
            slow = slow.next;
            fast = fast.next.next;

            if (slow == fast) {
                return true;
            }
        }
        return false;
    }
}
```