查阅更多的题解，请点击

Problem

142. Linked List Cycle II（Medium）

Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

To represent a cycle in the given linked list, we use an integer pos which represents the position (0-indexed) in the linked list where tail connects to. If pos is -1, then there is no cycle in the linked list.

Note: Do not modify the linked list.

Example 1:

```
Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the linked list, where tail connects to the second node.
```

Example 2:

```
Input: head = [1,2], pos = 0
Output: tail connects to node index 0
Explanation: There is a cycle in the linked list, where tail connects to the first node.
```

Example 3:

```
Input: head = [1], pos = -1
Output: no cycle
Explanation: There is no cycle in the linked list.
```

Solution

Corner Cases：

- 链表为空或仅有一个节点

O(n) time, O(1) space

首先利用**快慢指针**判断是否有环，有环时：此时快慢指针为相遇，设相遇点为A，慢指针走了N步，则快指针走了2N步；此时将快指针步长设为1，再指向头节点；两个指针继续前进，经过N步，显然会在A处相遇，由于步长相同，他们此时第一次相遇的位置就是环的入口（不一定为A）

详细分析可以看LeetCode 141的题解。

```
class Solution
{
public:
    ListNode *detectCycle(ListNode *head)
    {
        if (head == nullptr || head->next == nullptr)
            return nullptr;
        auto slow = head;
        auto fast = head;
        while (fast != nullptr && fast->next != nullptr)
        {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast)
                break;
        }
        if (slow != fast)
            return nullptr;
        fast = head;
        while (fast != slow)
        {
            slow = slow->next;
            fast = fast->next;
        }
        return fast;
    }
};
```

GitHub传送门