Problem 142: Linked List Cycle II

Problem Information

Difficulty: Medium
Acceptance Rate: 56.49%
Paid Only: No
Tags: Hash Table, Linked List, Two Pointers

Problem Description

Given the `head` of a linked list, return _the node where the cycle begins. If there is no cycle, return_`null`.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to (**0-indexed**). It is `-1` if there is no cycle. **Note that** `pos` **is not passed as a parameter**.

Do not modify the linked list.

Example 1:

![](https://assets.leetcode.com/uploads/2018/12/07/circularlinkedlist.png)

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:

![](https://assets.leetcode.com/uploads/2018/12/07/circularlinkedlist_test2.png)

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:

![](https://assets.leetcode.com/uploads/2018/12/07/circularlinkedlist_test3.png)

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

Constraints:

* The number of the nodes in the list is in the range `[0, 104]`. * `-105 <= Node.val <= 105` * `pos` is `-1` or a **valid index** in the linked-list.

Follow up: Can you solve it using `O(1)` (i.e. constant) memory?

Code Snippets

C++:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode(int x) : val(x), next(NULL) {}
* };
*/
class Solution {
public:
ListNode *detectCycle(ListNode *head) {

}
};
```

Java:

```
/**
* Definition for singly-linked list.
* class ListNode {
* int val;
* ListNode next;
* ListNode(int x) {
* val = x;
* next = null;
* }
* }
*/
public class Solution {
public ListNode detectCycle(ListNode head) {

}
}
```

Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, x):
# self.val = x
# self.next = None

class Solution:
def detectCycle(self, head: Optional[ListNode]) -> Optional[ListNode]:
```