141. Linked List Cycle

Easy 4919 638 Add to List Share

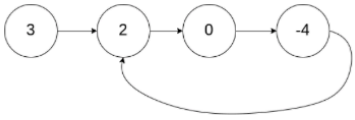
Easy

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

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. **Note that `pos` is not passed as a parameter.**

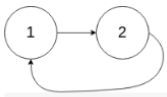
Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:




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

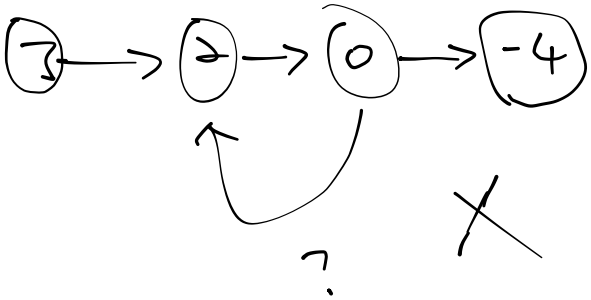
Example 2:




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

Example 3:


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



Since the Node 0 can only have one pointed next node.

∴ the cycle can only be formed through the tail node.

A. Use flag to label all visited nodes.

Failed to create outer flag for each

nodes.

Python3 · Autocomplete

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

class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        t=[]
        flag=0
        not_end = True
        while head and not_end:

            if flag in t :
                return True
            t.append(flag)
            flag+=1
            if head.next ==None:
                not_end = False
            head = head.next


        return False
```

Bug.

revisit will overwrite the flag.

To fix, Modif the node val directly after we visited it.

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

class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        while head: <=> head.next != None
            if head.val == None:
                return True
            head.val = None
            head = head.next
        return False
```

When we reach a node with val $\boxed{\text{None}}$

we found a cycle.

B. Use a set() to store visited node.

```
class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        visited = set()


        while head:
            if head in visited:
                return True
            else:
                visited.add(head)
                head = head.next
        return False
```

check

add into the visited pool after

jump to next node.

C. Floyd's cycle theory.

Check the solution in leetcode.

* If two ppl running inside a loop with diff speed,

they will meet eventually.