141. Linked List Cycle

Easy

1343116FavoriteShare

Given a linked list, determine if it has a cycle in it.

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 posis -1, then there is no cycle in the linked list.

Example 1:

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



Example 3:

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



Follow up:

Can you solve it using *O(1)* (i.e. constant) memory?

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

bool hasCycle(ListNode \*head) {

ListNode\* ptr1=head,\*ptr2=head;

while(ptr2!=NULL){

if(ptr2->next!=NULL){

ptr2=ptr2->next;

}else return false;

if(ptr2->next!=NULL){

ptr2=ptr2->next;

}else return false;

ptr1=ptr1->next;

if(ptr1==ptr2) return true;

}

return false;

}

};

Success

[Details](https://leetcode.com/submissions/detail/211376653/)

Runtime: 12 ms, faster than 100.00% of C++ online submissions for Linked List Cycle.

Memory Usage: 9.6 MB, less than 89.25% of C++ online submissions for Linked List Cycle.