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 pos is -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.



思路：

这就是非常简单的Linked list 循环问题， 就一条路，快慢指针

两个人都从head开始，快指针一次走两格，慢指针一次走一格

如果快指针指到Null，说明没有循环，

如果快指针与慢指针走到同一个点，说明有循环

/\*\*

\* Definition for singly-linked list.

\* class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) {

\* val = x;

\* next = null;

\* }

\* }

\*/

public class Solution {

public boolean hasCycle(ListNode head) {

ListNode fast=head;

ListNode slow=head;

while(fast!=null){

fast=fast.next;

if(fast!=null) fast=fast.next; else break;

//不能同时有两个next(fast=fast.next.next),因为如果下一个就是Null,就会out of index

slow=slow.next;

if(fast==slow) return true;

}

return false;

}

}