141. Linked List Cycle

Solved

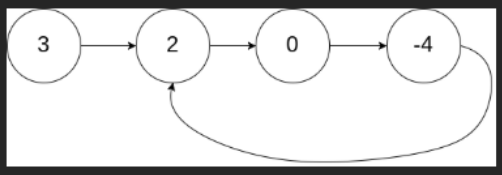
Easy | Topics | Companies

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).
```

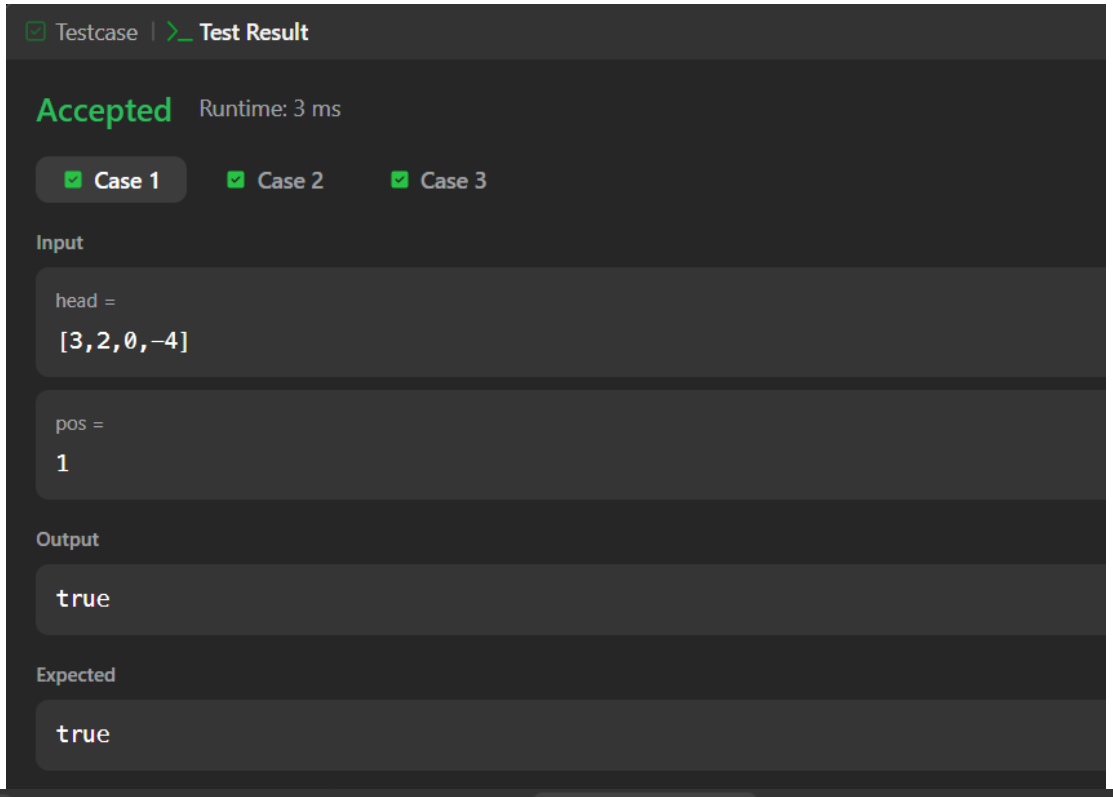
Code

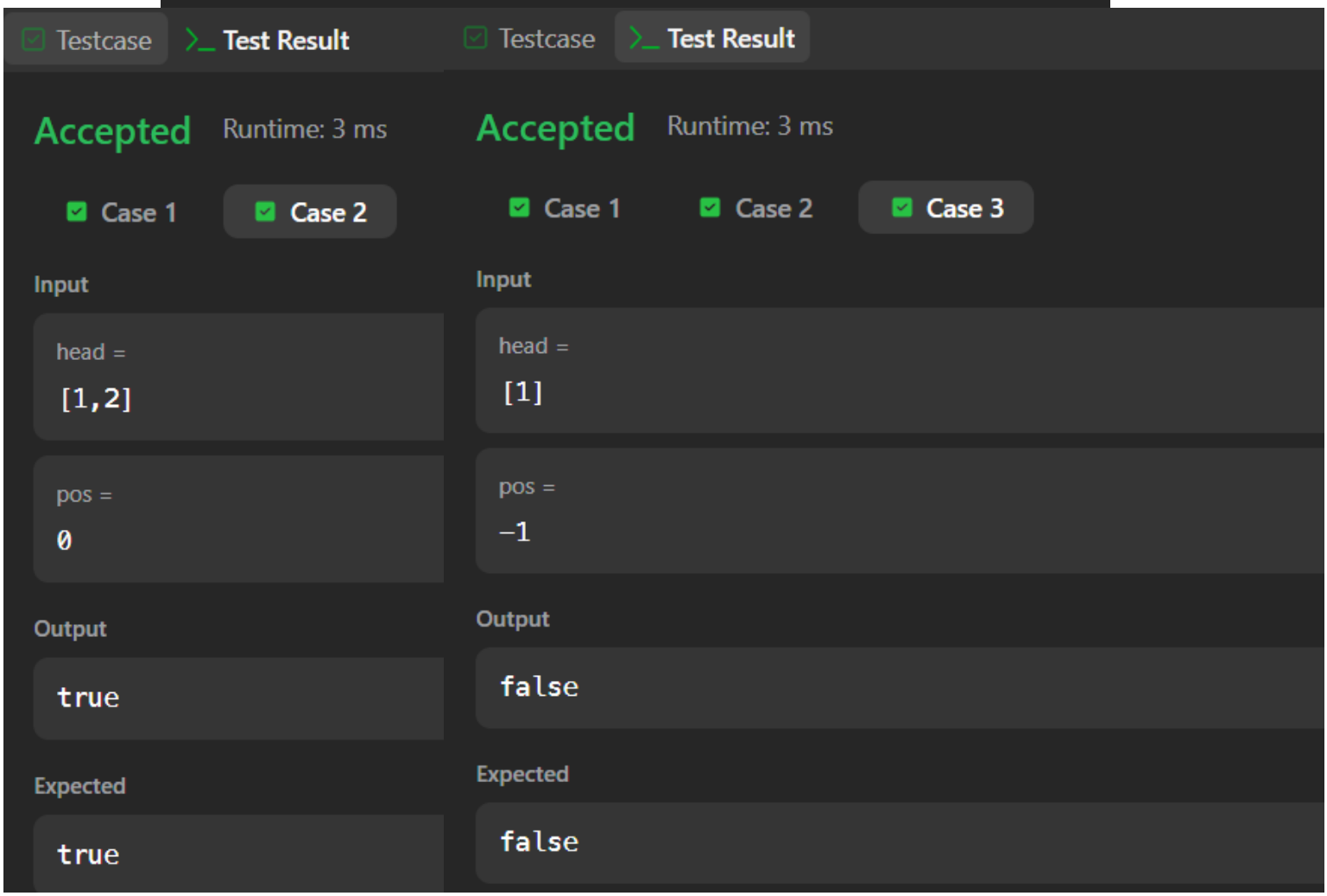
C | Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
bool hasCycle(struct ListNode *head) {
    struct ListNode * temp =head;
    struct ListNode * curr=head;

    while(curr !=NULL && curr->next!=NULL){
        temp=temp->next;
        curr=curr->next->next;

        if(temp==curr){
            return true;
        }
    }
    return false;
}
```


Testcase
Test Result
Accepted
Runtime: 3 ms
Case 1
Case 2
Case 3
Input
head =
[3,2,0,-4]
pos =
1
Output
true
Expected
true


Testcase
Test Result
Accepted
Runtime: 3 ms
Case 1
Case 2
Input
head =
[1,2]
pos =
0
Output
true
Expected
true
Testcase
Test Result
Accepted
Runtime: 3 ms
Case 1
Case 2
Case 3
Input
head =
[1]
pos =
-1
Output
false
Expected
false