Description | Editorial | Solutions (8.4K) | Submissic | Java | Auto

141. Linked List Cycle

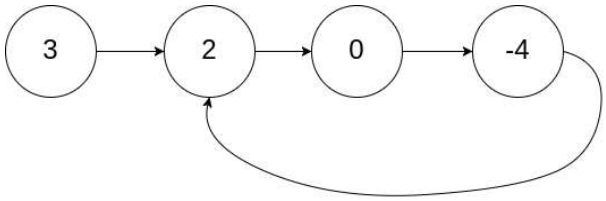
Easy | 13K | 1.1K | Companies

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**.

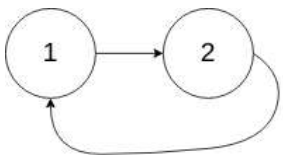
Return `true` *if there is a cycle in the linked list.* Otherwise, return `false`.

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).
```

Example 2:




```
 *          next = null;
 *      }
 * }
 */
public class Solution {
    public boolean hasCycle(ListNode he
        List<ListNode> al = new ArrayLi
    boolean flag = false;
    if (head != null) {
      ListNode current = head;
      while (current != null) {
        if (al.contains(current)) {
          flag = true;
          break;
        } else {
          al.add(current);
        }
        current = current.next;
      }
    }
    return flag;
    }
}
```

Console | Run | Subr