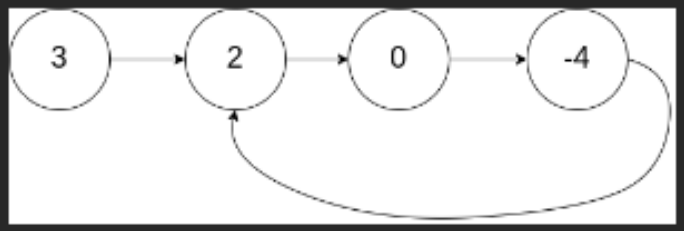
Detect the cycle in Linked List

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.**

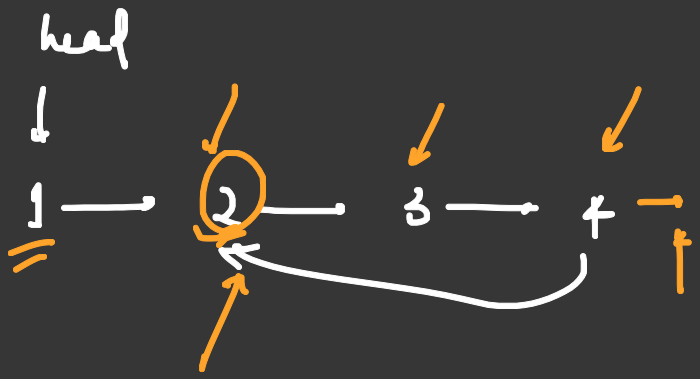
Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

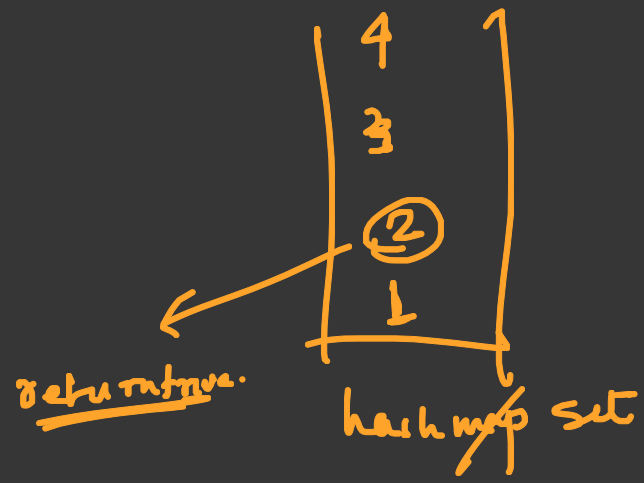
Example 1:

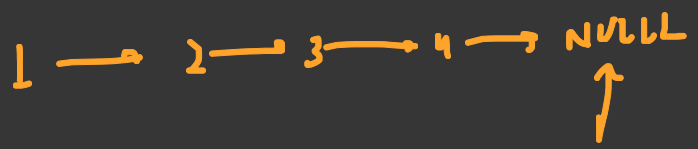


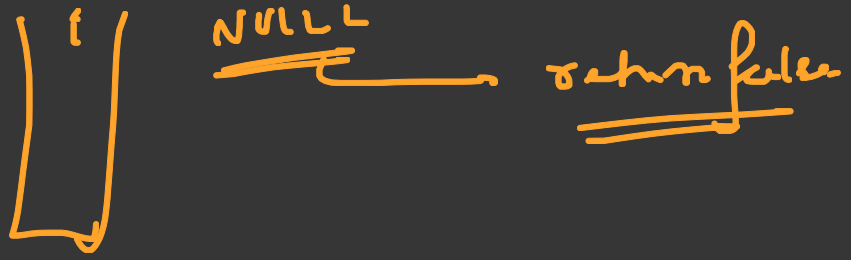
> **Input:** `head = [3,2,0,-4], pos = 1`
> **Output:** `true`
> **Explanation:** `There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).`

Brute force









- We use hashmap/hashset to save prev node to find cycle in it.

$$T.C. = O(N) \quad S.C. \sim O(N)$$

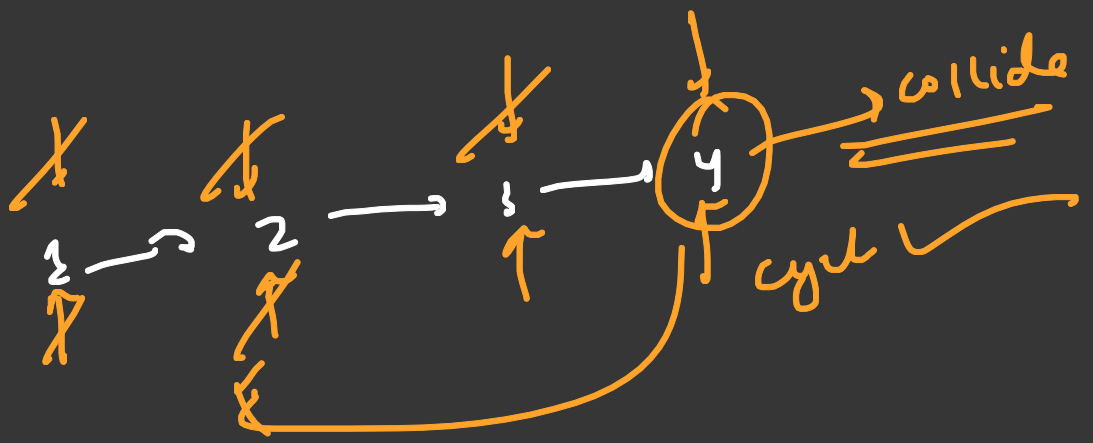
Optimal Approach:

Slow fast point method if the slow == fast → Error cycle.

edge not

slow = slow → next

fast = fast → next → next.



T.C = O(N)

S.C = O(1)