Check if Linked List Circular CodeStudio

Check if a LinkedList is a circular linked list or not.

Approach 1: Check if a given linked list is circular by traversing through the list.

In this approach, you traverse through the linked list, starting from the head, and keep moving until you either reach the end of the list or come back to the head node. If you come back to the head, then the list is circular; otherwise, it's not.

- **Time Complexity: O(N) - You need to visit each node once, where N is the number of nodes in the linked list.**
- **Space Complexity: O(1) - No additional data structures are used.**

Approach 2: Check if a given linked list is circular using a hash map to track visited nodes.

In this approach, you use a hash map to track the nodes you've visited. As you traverse through the linked list, you mark each visited node in the hash map. If you encounter a node that has already been marked, and it's the same as the head node, then the list is circular.

- **Time Complexity: O(N) - You need to visit each node once, where N is the number of nodes in the linked list.**
- **Space Complexity: O(N) - A hash map is used to store visited nodes.**

Approach 3: Check if a given linked list is circular using the Floyd's Tortoise and Hare algorithm (two-pointer approach).

This approach is based on the idea of two pointers moving through the list at different speeds. If the linked list is circular, the fast and slow pointers will eventually meet. If there is no cycle, the fast pointer will reach the end of the list.

- **Time Complexity: O(N) - In the worst case, the fast pointer will take N steps to reach the end or find a cycle.**
- **Space Complexity: O(1) - No additional data structures are used.**