Detect Loop in A Linked List LeetCode

Write a program to identify loop in a linked list

Approach 1: Hash Set (unordered_map) for Cycle Detection

- Traverse the linked list, marking each visited node within an unordered_map.
- If a node that has already been visited is encountered, a cycle is detected.
- This approach requires additional memory to store visited nodes.

Time Complexity: O(n), where n is the number of nodes in the linked list.

Space Complexity: O(n), attributed to the space consumed by the hash map to store visited nodes.

Approach 2: Floyd's Cycle Detection Algorithm (Tortoise and Hare)

- Use two pointers, slow and fast, to navigate the linked list.
- The slow pointer advances one node at a time, while the fast pointer progresses by two nodes at a time.
- If a cycle exists, the two pointers will eventually meet within the cycle.

Time Complexity: O(n), where n represents the number of nodes in the linked list.

Space Complexity: O(1), as only a fixed amount of additional space is used.