Detect Loop’s Starting Point in a Linked List LeetCode

Write a program to identify loop in a linked list and return the starting point of the linked list.

Approach 1: Hash Set (unordered_map) for Cycle Detection

- A hash map is used to store visited nodes.
- Traverse the linked list while marking each visited node in the hash map.
- If a node is encountered that has already been visited, a cycle is detected.

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

Space Complexity: O(n), due to the hash map used for storing visited nodes.

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

- Two pointers, slow and fast (Tortoise and Hare), are used to traverse the linked list.
- The slow pointer moves one step at a time, while the fast pointer moves two steps at a time.
- If a cycle exists, the two pointers will eventually meet inside the cycle.
- After meeting, one pointer is reset to the head of the linked list, and both pointers move one step at a time until they meet again. The meeting point is the start of the cycle.

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

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