Documentation for Partitioning a Linked List

Problem Statement

Given the head of a linked list and a value x, partition the list such that all nodes with values less than x come before nodes with values greater than or equal to x. The relative order of the nodes in each partition should be preserved.

Example 1

Input:

head = [1, 4, 3, 2, 5, 2]

x = 3

Output: [1, 2, 2, 4, 3, 5]

Example 2

Input:

head = [2, 1]

x = 2

Output: [1, 2]

Constraints

- The number of nodes in the list is in the range [0, 200].
- Node values are between -100 and 100.
- x is between -200 and 200.

Approach

The solution involves creating two separate lists: one for nodes with values less than x and another for nodes with values greater than or equal to x. Then, these lists are merged to form the desired partitioned list.

1. Initialization:

- Create two dummy nodes, less_head and greater_head, which will serve as the heads of the two lists.
- Initialize two pointers, less and greater, to point to the dummy nodes.

2. Partitioning:

- *Traverse the original list. For each node:*
  - ✓ If the node's value is less than x, append it to the less list.
  - ✓ Otherwise, append it to the greater list.
  - ✓ Move the pointers (less and greater) forward as nodes are appended to ensure the lists are built correctly.

3. Combining Lists:

- After the traversal, connect the less list to the greater list by setting less.next to greater_head.next.
- Ensure the last node of the greater list points to None to terminate the list.

4. Return Result:

- The head of the new partitioned list is the next node of less_head.

Explanation of the Code

1. **Class Definition (ListNode):**

- Defines a node in the singly linked list with attributes val (value of the node) and next (pointer to the next node).

2. **Solution Class (Solution):**

- Defines the method partition which takes the head of the linked list and the integer x as parameters.
- Creates dummy nodes less_head and greater_head to start the two partitions.
- Uses two pointers less and greater to build the partitions by traversing the original list.
- Connects the two partitions and ensures the end of the greater list points to None.
- Returns the head of the new partitioned list, which is less_head.next.

Usage

To use this solution, create an instance of Solution and call the partition method with the head of your linked list and the partition value x.

Notes

- The solution maintains the relative order of nodes in both partitions.
- Dummy nodes (less_head and greater_head) simplify handling edge cases and merging lists.
- The time complexity is O(n), where n is the number of nodes in the linked list, as it involves a single pass through the list. The space complexity is O(1) additional space, as it uses a constant number of pointers.