Documentation: Deep Copy of a Linked List with Random Pointers

Problem Description

You are given a linked list of length n where each node contains an additional random pointer that can point to any node in the list, or null. Your task is to construct a deep copy of the list. The deep copy should consist of exactly n brand-new nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointers of the new nodes should point to new nodes in the copied list, such that the pointers in the original list and copied list represent the same list state. None of the pointers in the new list should point to nodes in the original list.

Input/Output Format

Input:

The linked list is represented as a list of n nodes. Each node is represented as a pair [val, random_index] where:

- **val:** An integer representing the value of the node.
- **random_index:** The index of the node (ranging from 0 to n-1) that the random pointer points to, or null if it does not point to any node.

Output:

A list of n nodes in the copied linked list, where each node is represented as a pair [val, random_index] with the same structure as the input.

Example 1:

- **Input:** head = [[7,null],[13,0],[11,4],[10,2],[1,0]]
- **Output:** [[7,null],[13,0],[11,4],[10,2],[1,0]]

Example 2:

- **Input:** head = [[1,1],[2,1]]
- **Output:** [[1,1],[2,1]]

Example 3:

- **Input:** head = [[3,null],[3,0],[3,null]]
- **Output:** [[3,null],[3,0],[3,null]]

Constraints

- 0 <= n <= 1000
- -10^4 <= Node.val <= 10^4
- Node.random is either null or points to some node in the linked list.

Approach

1. **Interweaving Original and Copied Nodes:**

- Traverse the original linked list. For each node, create a new node with the same value. Insert this new node immediately after the original node.
- For example, if the original list is A -> B -> C, after this step it will become A -> A' -> B -> B' -> C -> C', where A', B', and C' are new nodes.

2. **Assign Random Pointers to New Nodes:**
   - Traverse the modified list. For each original node, set the random pointer of its corresponding new node to point to the new node that follows the original node's random pointer.

3. **Separate the Original and Copied Lists:**
   - Traverse the list to separate the original and copied nodes. Restore the original list and extract the copied list.
   - Set the next pointer of each original node to skip the copied nodes, and set the next pointer of each copied node to the next copied node.

This approach ensures that the new list is a deep copy of the original, maintaining the structure and relationships defined by the next and random pointers.