Daily Coding Problem #131

Problem

This question was asked by Snapchat.

Given the head to a singly linked list, where each node also has a “random” pointer that points to anywhere in the linked list, deep clone the list.

Solution

This problem has a straightforward solution using O(n) space:

- Create a clone of the linked list, disregarding `random` pointers.
- Make a hashmap that maps from an original node to its cloned counterpart.
- Iterate through both the clone and originals at the same time. For a given clone node, find the original's `random` clone counterpart in the hashmap, and set it as its `random` node.

However, there's a clever way to use even less space.

- First, double the linked list by interleaving it with cloned nodes (without `random` set). For example, given 1 -> 2 -> 3, becomes 1 -> 1 -> 2 -> 2 -> 3 -> 3.
- Set the cloned nodes' `random` by following the original, previous node's `random.next`.
- Restore the linked lists by separating them. For example, each original nodes need to set `node.next = node.next.next`.

```
def clone(node):
    node = double(node)
    set_random_pointers(node)

    clone_head = node.next

    while node:
        clone_match = node.next

        if clone_match.next:
            node.next, clone_match.next = node.next.next, clone_match.next.next
        else:
            node.next, clone_match.next = node.next.next, None

        node = node.next

    return clone_head

def set_random_pointers(node):
    while node:
```

```python
        clone_match = node.next
        clone_match.random = node.random.next

        node = node.next.next

def double(node):
    root = node
    while node:
        copy = Node(node.val)
        next = node.next

        node.next = copy
        copy.next = next
        node = next

    return root
```

Since we only store pointers, this only takes O(1) extra space.