Problem Name: Merge k Sorted Lists

Link: https://leetcode.com/problems/merge-k-sorted-lists/description/

Solution Language(s): Java

Approach

For this problem, my approach is a little complex. I began by deciding to place all the values between all of the linked lists into a HashMap, where the keys represented the integer in the linked lists, and the values represented the occurrences of those integers. This would allow me to then loop through all of the keys, from smallest to largest, and then add nodes to the resulting linked lists based on the associated value pairing.

Solution

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next;
}
 * }
 */
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        ListNode head = new ListNode();
        ListNode temp = head;
        HashMap<Integer, Integer> table = new HashMap<>();
        for (int i = 0; i < lists.length; i++) {
            ListNode curr = lists[i];
            while (curr != null) {
                if (table.containsKey(curr.val)) {
                    table.replace(curr.val, table.get(curr.val) + 1);
                } else {
                    table.put(curr.val, 1);
                }
                curr = curr.next;
            }
        }
        Set<Integer> keys = table.keySet();
        ArrayList<Integer> keys2 = new ArrayList<>(keys);
        Collections.sort(keys2);
        for (int j = 0; j < keys2.size(); j++) {
            int key = keys2.get(j);
            int len = table.get(key);
            for (int i = 0; i < len; i++) {
                temp.next = new ListNode(key);
                temp = temp.next;
            }
        }
        return head.next;
    }
}
```

Time Complexity: O(nlog(n))

In my solution to this problem, the time complexity is O(nlog(n)), where n represents the total nodes between all the linked lists.

Space Complexity: O(n)

The space complexity for my solution is O(n), as all of the nodes are stored as keys in the hash map, thus yielding n, and the HashMap's values are then added to a linked list that is returned, thus again involving n, and simplifying to a space complexity of O(n).

Conclusion

In conclusion, this problem is all about how you can compare across k lists. Doing so using only loops and comparisons would yield an insanely large time complexity, as every value is being checked across the values of the other lists. My solution to this problem was to have one loop go through every node and place the integers into a HashMap for quick retrieval. Although