1171. Remove Zero Sum Consecutive Nodes from Linked List

Given the head of a linked list, we repeatedly delete consecutive sequences of nodes that sum to 0 until there are no such sequences.

After doing so, return the head of the final linked list. You may return any such answer.

(Note that in the examples below, all sequences are serializations of ListNode objects.)

Example 1:

```
Input: head = [1,2,-3,3,1]
Output: [3,1]
Note: The answer [1,2,1] would also be accepted.
```

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public ListNode removeZeroSumSublists(ListNode head) {

        // two pointers (O(n^2) time, O(1) space)
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode cur = dummy;

        while(cur != null) {
            int sum = 0;
            while(head != null) {
                sum += head.val;
                if (sum == 0) {
                    cur.next = head.next;
                }
                head = head.next;
            }
            cur = cur.next;
            if (cur != null)
                head = cur.next;
        }

        return dummy.next;
    }
}
```

```
class Solution {
    public ListNode removeZeroSumSublists(ListNode head) {

        // prefix sum with hashmap, O(N) time with O(N) space
        ListNode dummy = new ListNode(0);
        ListNode curr = dummy;
        Map<Integer, ListNode> map = new HashMap<>();
        dummy.next = head;
        int prefixSum = 0;

        while (curr != null) {
            prefixSum += curr.val;
            if (map.containsKey(prefixSum)) {
                ListNode node = map.get(prefixSum).next;
                int sum = prefixSum;
                while (node != curr) {
                    sum += node.val;
                    map.remove(sum);
                    node = node.next;
                }
                map.get(prefixSum).next = curr.next;
            } else {
                map.put(prefixSum, curr);
            }
            curr = curr.next;
        }

        return dummy.next;
    }
}
```