2181. Merge Nodes in Between Zeros

Solved ✓

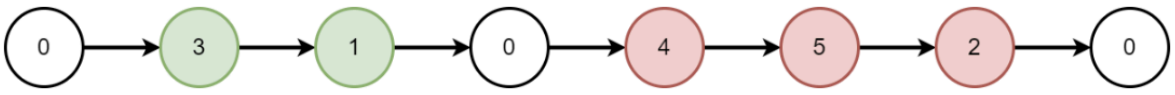
Medium Topics Companies Hint

You are given the `head` of a linked list, which contains a series of integers **separated** by `0`'s. The **beginning** and **end** of the linked list will have `Node.val == 0`.

For **every** two consecutive `0`'s, **merge** all the nodes lying in between them into a single node whose value is the **sum** of all the merged nodes. The modified list should not contain any `0`'s.

Return *the* `head` *of the modified linked list*.

Example 1:



```
Input: head = [0,3,1,0,4,5,2,0]
Output: [4,11]
Explanation:
The above figure represents the given linked list. The modified list
contains
- The sum of the nodes marked in green: 3 + 1 = 4.
- The sum of the nodes marked in red: 4 + 5 + 2 = 11.
```

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     public var val: Int
 *     public var next: ListNode?
 *     public init() { self.val = 0; self.next = nil; }
 *     public init(_ val: Int) { self.val = val; self.next = nil; }
 *     public init(_ val: Int, _ next: ListNode?) { self.val = val;
self.next = next; }
 * }
 */
class Solution {
    func mergeNodes(_ head: ListNode?) -> ListNode? {
        var arr:[Int] = []

        var temp = head
        var t = 0
        while(temp != nil){
            if(temp!.val != 0 ){
                t = t + temp!.val
            } else {
                if(t != 0) {
                    arr.append(t)
                    t = 0
                }
            }
            temp = temp?.next
        }

        // print(arr)

        var output = ListNode()
        let tt = output
        for i in arr {
            let n = ListNode(i)
            output.next = n
            output = output.next!
        }

        return tt.next
    }
}
```