725. Split Linked List in Parts

Solved

Medium | Topics | Companies | Hint

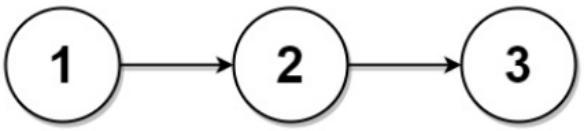
Given the `head` of a singly linked list and an integer `k`, split the linked list into `k` consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return *an array of the* `k` *parts.*

Example 1:




```
Input: head = [1,2,3], k = 5
Output: [[1],[2],[3],[],[]]
Explanation:
```

Code

```
class Solution {
    func splitListToParts(_ head: ListNode?, _ k: Int) ->
[ListNode?] {
        guard var r = head else {
            return Array(repeating:ListNode(0).next, count:k)
        }

        var root = head
        var i = 1
        while(root?.next != nil) {
            i = i + 1
            root = root?.next
        }
```

```
        var arr:[Int] = Array(repeating:i/k, count:k)
        for i in 0..<i%k {
            arr[i] = arr[i] + 1
        }

        let dum = ListNode(0)
        var j = 0
        var index = 1
        root = head
        var ans:[ListNode?] = []
        for i in arr {
            print(i)
            if(i == 0){
                ans.append(dum.next)
                continue
            }
            var k = 0
            var t:ListNode? = ListNode(0)
            t?.next = root
            ans.append(t?.next)
            while(k<i) {
                root = root?.next
                t = t?.next
                k = k + 1
            }
            t?.next = nil
        }
        return ans
    }
}
```