Problem 3063: Linked List Frequency

Problem Information

Difficulty: Easy
Acceptance Rate: 85.35%
Paid Only: Yes
Tags: Hash Table, Linked List, Counting

Problem Description

Given the `head` of a linked list containing `k` **distinct** elements, return _the head to a linked list of length_`k` _containing the frequency of each **distinct** element in the given linked list in **any order**._

Example 1:

Input: head = [1,1,2,1,2,3]

Output: [3,2,1]

Explanation: There are `3` distinct elements in the list. The frequency of `1` is `3`, the frequency of `2` is `2` and the frequency of `3` is `1`. Hence, we return `3 -> 2 -> 1`.

Note that `1 -> 2 -> 3`, `1 -> 3 -> 2`, `2 -> 1 -> 3`, `2 -> 3 -> 1`, and `3 -> 1 -> 2` are also valid answers.

Example 2:

Input: head = [1,1,2,2,2]

Output: [2,3]

Explanation: There are `2` distinct elements in the list. The frequency of `1` is `2` and the frequency of `2` is `3`. Hence, we return `2 -> 3`.

Example 3:

Input: head = [6,5,4,3,2,1]

Output: [1,1,1,1,1,1]

Explanation: There are `6` distinct elements in the list. The frequency of each of them is `1`. Hence, we return `1 -> 1 -> 1 -> 1 -> 1 -> 1`.

Constraints:

* The number of nodes in the list is in the range `[1, 105]`. * `1 <= Node.val <= 105`

Code Snippets

C++:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode() : val(0), next(nullptr) {}
* ListNode(int x) : val(x), next(nullptr) {}
* ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
ListNode* frequenciesOfElements(ListNode* head) {

}
};
```

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 frequenciesOfElements(ListNode head) {

}
}
```

Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution:
def frequenciesOfElements(self, head: Optional[ListNode]) ->
Optional[ListNode]:
```