Problem 817: Linked List Components

Problem Information

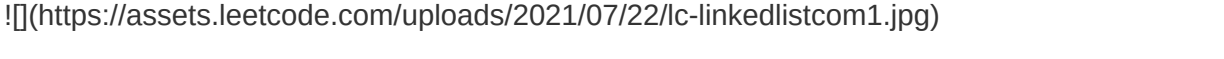
Difficulty: Medium
Acceptance Rate: 57.54%
Paid Only: No
Tags: Array, Hash Table, Linked List

Problem Description

You are given the `head` of a linked list containing unique integer values and an integer array `nums` that is a subset of the linked list values.

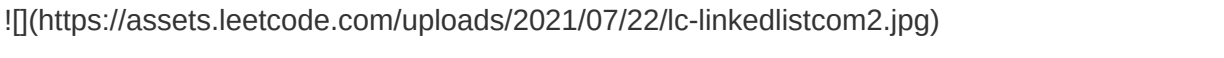
Return _the number of connected components in_`nums` _where two values are connected if they appear**consecutively** in the linked list_.

Example 1:



Input: head = [0,1,2,3], nums = [0,1,3] **Output:** 2 **Explanation:** 0 and 1 are connected, so [0, 1] and [3] are the two connected components.

Example 2:



Input: head = [0,1,2,3,4], nums = [0,3,1,4] **Output:** 2 **Explanation:** 0 and 1 are connected, 3 and 4 are connected, so [0, 1] and [3, 4] are the two connected components.

Constraints:

* The number of nodes in the linked list is `n`. * `1 <= n <= 104` * `0 <= Node.val < n` * All the values `Node.val` are **unique**. * `1 <= nums.length <= n` * `0 <= nums[i] < n` * All the values of `nums` are **unique**.

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:
int numComponents(ListNode* head, vector<int>& nums) {

}
};
```

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 int numComponents(ListNode head, int[] nums) {

}
}
```

Python3:

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