给定一个链表（链表结点包含一个整型值）的头结点 `head`。

同时给定列表 `G`，该列表是上述链表中整型值的一个子集。

返回列表 `G` 中组件的个数，这里对组件的定义为：链表中一段最长连续结点的值（该值必须在列表 `G` 中）构成的集合。

示例 1：

输入：

```
head: 0->1->2->3
G = [0, 1, 3]
```

输出： 2

解释：

链表中,`0` 和 `1` 是相连接的，且 `G` 中不包含 `2`，所以 `[0, 1]` 是 `G` 的一个组件，同理 `[3]` 也是一个组件，故返回 `2`。

示例 2：

输入：

```
head: 0->1->2->3->4
G = [0, 3, 1, 4]
```

输出： 2

解释：

链表中，`0` 和 `1` 是相连接的，`3` 和 `4` 是相连接的，所以 `[0, 1]` 和 `[3, 4]` 是两个组件，故返回 `2`。

注意:

- 如果 `N` 是给定链表 `head` 的长度，`1 <= N <= 10000`。
- 链表中每个结点的值所在范围为 `[0, N - 1]`。
- `1 <= G.length <= 10000`
- `G` 是链表中所有结点的值的一个子集.

分析，该题是找包含在链表内连续的个数（有毒，个人没看清，以为是最长的连续个数），所以把连续的统计出来，当不连续后，判断之前有没有连续的，有则数目+1，然后连续数目置零。

代码

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
```

```
 * };
 */
class Solution {
public:
    int numComponents(ListNode* head, vector<int>& G) {
        map<int,bool> bj;
        int nums=0;
        int num=0;
        for(int i=0;i<G.size();i++){
            bj[G[i]]=true;
        }
        while(head!=NULL){
            if(bj[head->val]){
                num++;
            }else{
                if(num>0){
                    nums++;
                }
                num=0;
            }
            head=head->next;
        }
        if(num>0){
            nums++;
        }
        return nums;
    }
};
```