给定一个排序链表，删除所有含有重复数字的节点，只保留原始链表中 *没有重复出现* 的数字。

示例 1:

输入： 1->2->3->3->4->4->5
输出： 1->2->5

示例 2:

输入： 1->1->1->2->3
输出： 2->3

解法一，递归解法，直接遍历到链表最后面，动态的统计数字出现个数，若是出现个数大于1，则返回本节点之后的节点，若等于1，则返回本节点

代码

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* dfs(map<int,int> &bj,ListNode* head){
        if(head==NULL){
            return head;
        }
        bj[head->val]++;
        if(head->next!=NULL){
            head->next=dfs(bj,head->next);
        }
        if(bj[head->val]==1){
            return head;
        }
        return head->next;
    }
    ListNode* deleteDuplicates(ListNode* head) {
        map<int,int> bj;
        return dfs(bj,head);
    }
};
```

解法2，遍历数组，第一遍统计个数，第二遍删除重复元素。

代码 （个人菜鸡，写的好复杂啊）

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(head==NULL){
            return head;
        }
        map<int,int> bj;
        ListNode* node=head;
        while(node!=NULL){
            bj[node->val]++;
            node=node->next;
        }
        node=head;
        ListNode* node1=NULL;
        head=NULL;
        while(node!=NULL){
            if(bj[node->val]==1){
                node1=node;
                head=node;
                node=node->next;
                break;
            }
            node=node->next;
        }
        while(node!=NULL){
            if(bj[node->val]==1){
                node1->next=node;
                node1=node1->next;
            }
            node=node->next;
        }
        if(node1!=NULL){
            node1->next=NULL;
        }
        return head;
    }
};
```