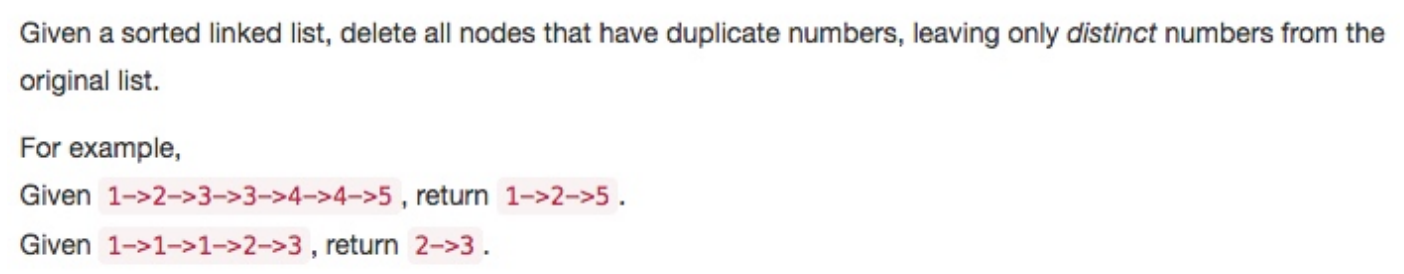
082. Remove Duplicates from Sorted List II

082 Remove Duplicates from Sorted List II

- **Linked List**

Description



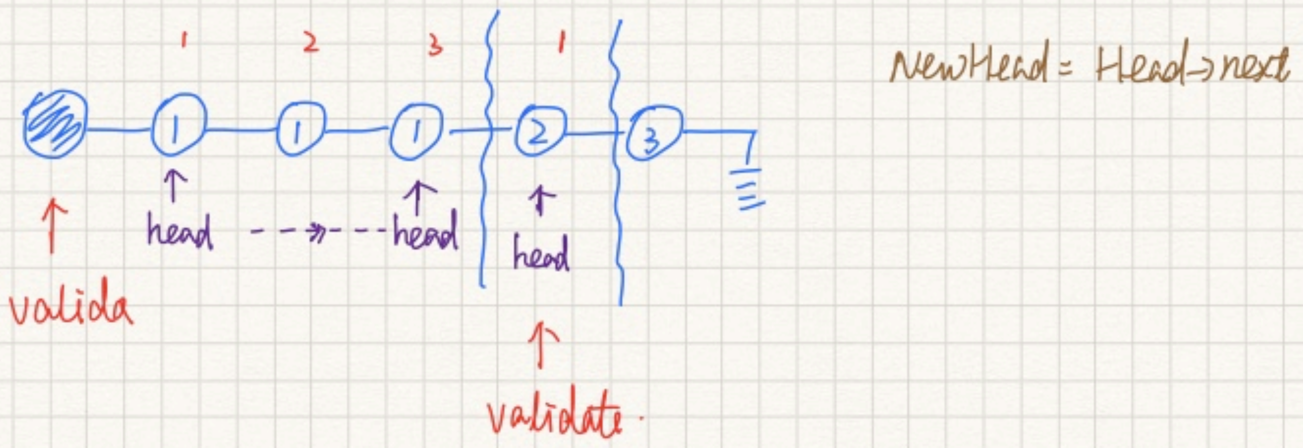
Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only *distinct* numbers from the original list.

For example,
Given `1->2->3->3->4->4->5`, return `1->2->5`.
Given `1->1->1->2->3`, return `2->3`.

1. Thought line

To each "distinct element checking loop":

- Count times : countTimes
- If countTimes ==1 ;
  - validatePointer pointers to this element
  - Next "distinct element checking loop":
- If counterTimes >1 ;
  - validatePointer stay
  - Next "distinct element checking loop":



2. Linked List

```cpp
/**
 * 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) {
        ListNode* dummyHead = new ListNode(0);
        dummyHead->next = head;
        ListNode* validatedPtr = dummyHead;
        int actElement = 0;
        while(head!=nullptr){
            ++actElement;
            while(head->next!=nullptr && head->next->val == head->val){
                ++actElement;
                head = head->next;
            }
            ListNode* newHead = head->next;
            if (actElement==1){
                validatedPtr->next = head;
                validatedPtr = validatedPtr->next;
            }
            else{
                validatedPtr->next = nullptr;
                head->next = nullptr;
            }
            head = newHead;
            actElement = 0;
        }
        return dummyHead->next;
    }
};




/**
 * 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) {
        ListNode* dummyHead = new ListNode(0);
        dummyHead->next = head;
        ListNode* validatedPtr = dummyHead;
        int actElement = 0;
        while(head!=nullptr){
            ++actElement;
            while(head->next!=nullptr && head->next->val == head->val){
                ++actElement;
                head = head->next;
            }
            if (actElement==1){
                validatedPtr->next = head;
                validatedPtr = validatedPtr->next;
            }
            /*
                @_: if actElement >1, validatedPtr doesn't move
                @_: dummyHead list is dominated by dummyHead and validatedPtr ONLY.

            */
            head = head->next;
            actElement = 0;
        }
        validatedPtr->next = nullptr;
        return dummyHead->next;
    }
};
```