一、题目说明

题目148. Sort List，对链表进行排序，时间复杂度要求是O(nlog(n))，空间复杂度要求是常量。难度是Medium！

二、我的解答

根据要求，唯一符合标准的是归并排序。

```
class Solution{
    public:
        ListNode* sortList(ListNode* head){
            if(head==NULL || head->next==NULL) return head;
            return mergeSort(head);
        }
        //归并排序
        ListNode* mergeSort(ListNode* node){
            if(node==NULL || node->next==NULL) return node;
            ListNode* fast= node,*slow = node;
            ListNode* breakNode = node;//breakNode 指向l1的最后一个元素
            //也可以采用先遍历一遍，统计链表节点的数量，然后归并排序
            //找到链表中间
            while(fast && fast->next){
                fast = fast->next->next;
                breakNode = slow;
                slow = slow->next;
            }
            breakNode->next = NULL;
            ListNode* l1 = mergeSort(node);
            ListNode* l2 = mergeSort(slow);

            return merge(l1,l2);
        }

        //合并两个链表 recursive
        ListNode* merge(ListNode* l1,ListNode* l2){
            if(l1 == NULL){
                return l2;
            }
            if(l2 == NULL){
                return l1;
            }
            if(l1->val < l2->val){
                l1->next = merge(l1->next,l2);
                return l1;
            }else{
                l2->next = merge(l2->next,l1);
                return l2;
            }
        }
};
```

性能如下：

```
Runtime: 60 ms, faster than 41.32% of C++ online submissions for Sort List.
Memory Usage: 16.2 MB, less than 15.00% of C++ online submissions for Sort List.
```

三、优化措施

将merge函数，修改为非递归版本：

```
class Solution{
    public:
        ListNode* sortList(ListNode* head){
            if(head==NULL || head->next==NULL) return head;
            return mergeSort(head);
        }
        //归并排序
        ListNode* mergeSort(ListNode* node){
            if(node==NULL || node->next==NULL) return node;
            ListNode* fast= node,*slow = node;
            ListNode* breakNode = node;//breakNode 指向l1的最后一个元素
            //找到链表中间
            while(fast && fast->next){
                fast = fast->next->next;
                breakNode = slow;
                slow = slow->next;
            }
            breakNode->next = NULL;
            ListNode* l1 = mergeSort(node);
            ListNode* l2 = mergeSort(slow);

            return merge(l1,l2);
        }
        //合并两个链表
        ListNode* merge(ListNode* l1,ListNode* l2){
            if(l1 == NULL) return l2;
            if(l2 == NULL) return l1;
            ListNode dummy(0);
            ListNode* p = &dummy;
            while(l1!=NULL && l2!=NULL){
                if(l1->val < l2->val){
                    p->next = l1;
                    l1 = l1->next;
                }else{
                    p->next = l2;
                    l2 = l2->next;
                }
                p = p->next;
            }
            if(l1 !=NULL){
                p->next = l1;
            }
            if(l2 !=NULL){
                p->next = l2;
            }
            return dummy.next;
        }
};
```

```
Runtime: 52 ms, faster than 67.44% of C++ online submissions for Sort List.
Memory Usage: 15 MB, less than 15.00% of C++ online submissions for Sort List.
```