1. Sort List

Sort a linked list in *O*(*n* log *n*) time using constant space complexity.

**Example 1:**

Input: 4->2->1->3  
Output: 1->2->3->4

**Example 2:**

Input: -1->5->3->4->0  
Output: -1->0->3->4->5

**解法1** 归并排序。用两个函数实现：

* merge：将两个有序链表合在一起
* merge\_sort：将无序链表排序

找中间位置用双指针

/\*\*  
 \* 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:  
 ListNode\* sortList(ListNode\* head) {  
 if(head == NULL || head->next == NULL)return head;  
 ListNode \*mid, \*pre;  
 find\_mid(head, mid, pre);  
 pre->next = NULL;  
 return merge(sortList(head), sortList(mid));  
 }  
 // 找中点时，需要对mid和pre做修改，因此需要传引用  
 void find\_mid(ListNode\* s, ListNode\* &mid, ListNode\* &pre){  
 ListNode \*pp = s;  
 mid = s;  
 while(pp && pp->next){  
 pre = mid;  
 mid = mid->next;  
 pp = pp->next->next;  
 }  
 }  
 ListNode\* merge(ListNode\* s1, ListNode\* s2){  
 ListNode \*head = new ListNode, \*p = s1, \*q = s2;  
 ListNode \*cur = head;  
 while(p != NULL && q != NULL){  
 if(p->val < q->val){  
 cur->next = p;  
 p = p->next;  
 }else{  
 cur->next = q;  
 q = q->next;  
 }  
 cur = cur->next;  
 }  
 if(p != NULL)cur->next = p;  
 else cur->next = q;  
 return head->next;  
 }  
};

**解法2** 快速排序。partition过程中，可以用两个链表small和large分别存储pivot左侧和右侧的数据

class Solution {  
public:  
 ListNode\* sortList(ListNode\* head) {  
 if(!head || !head->next) return head;  
   
 ListNode\* cur = head->next;  
 ListNode\* small = new ListNode(0);  
 ListNode\* large = new ListNode(0);  
 ListNode\* sp = small;  
 ListNode\* lp = large;  
 // partition  
 while(cur){  
 if(cur->val<head->val){  
 sp->next = cur;  
 sp = cur;  
 }  
 else{  
 lp->next = cur;  
 lp = cur;  
 }  
 cur = cur->next;  
 }  
 sp->next = NULL;  
 lp->next = NULL;  
 small=sortList(small->next);  
 large=sortList(large->next);  
 cur = small;  
 if(cur){  
 while(cur->next) cur = cur->next;  
 cur->next = head;  
 head->next = large;  
 return small;  
 }else{  
 head->next = large;  
 return head;  
 }  
 }  
};