148. Sort List

Medium

119463FavoriteShare

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

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

ListNode\* merge(ListNode\* ptr1, ListNode\* ptr2){

ListNode\* dummy = new ListNode(0);

ListNode\* mover = dummy;

while(ptr1&&ptr2){

if(ptr1->val<ptr2->val){

mover->next = ptr1;

ptr1=ptr1->next;

mover=mover->next;

}else{

mover->next = ptr2;

ptr2=ptr2->next;

mover=mover->next;

}

}

while(ptr1){

mover->next=ptr1;

ptr1=ptr1->next;

mover=mover->next;

}

while(ptr2){

mover->next=ptr2;

ptr2=ptr2->next;

mover=mover->next;

}

return dummy->next;

}

ListNode\* sortList(ListNode\* head) {

if(head==NULL||head->next==NULL) return head;

//cout<<head->val<<endl;

ListNode\*ptr1=head,\*ptr2=head->next;

while(ptr2&&ptr2->next!=NULL){

ptr2=ptr2->next;

ptr1=ptr1->next;

ptr2=ptr2->next;

}

ListNode\* head2=ptr1->next;

ptr1->next=NULL;

return merge(sortList(head),sortList(head2));

}

};

Success

[Details](https://leetcode.com/submissions/detail/211395026/)

Runtime: 56 ms, faster than 67.33% of C++ online submissions for Sort List.

Memory Usage: 25 MB, less than 18.41% of C++ online submissions for Sort List.