分析如下：  
挺有意思的一道题目。表面上看，能够有O(n lgn)时间复杂度的算法为，快速排序，堆排序，归并排序，三者的空间复杂度分别为O(1), O(N),O(N)  
所以一开始，我想着用快速排序的方法来解决，但是发现代码很难写出来。于是网上看了一下提示，发现其实方法选错了。应该使用的方法是归并排序。

通常而言，也就是针对数组而言，归并排序的空间复杂度为O(N), 你需要开出O(N)的额外空间来容纳数组，来表示归并后的顺序。但是，对于链表而言，你可以省下这部分空间的开销，你只需要改变节点的next指针的指向，就可以表示新的归并后的顺序了，所以空间复杂度陡然降到了O(1)。

小结扩展：

快速排序和归并排序的时间复杂度都是O(N lgN)，但是CLRS说了，实践证明快速排序的速度比归并排序的速度更快，为什么呢?另外其实这个结论是有限制范围的，当对数组进行排序的时候，这个结论适用。为什么对于链表，却是归并排序的速度优于快速排序呢？[这里](http://stackoverflow.com/questions/7629904/why-is-mergesort-better-for-linked-lists)看到的一段对比说得挺好，直接抄过来。

One of the main sources of efficiency in quicksort is locality of reference, where the computer hardware is optimized so that accessing memory locations that are near one another tends to be faster than accessing memory locations scattered throughout memory. The partitioning step in quicksort typically has excellent locality, since it accesses consecutive array elements near the front and the back. As a result, quicksort tends to perform much better than other sorting algorithms like heapsort even though it often does roughly the same number of comparisons and swaps, since in the case of heapsort the accesses are more scattered.  
  
Additionally, quicksort is typically much faster than other sorting algorithms because it operates in-place, without needing to create any auxiliary arrays to hold temporary values. Compared to something like merge sort, this can be a huge advantage because the time required to allocate and deallocate the auxiliary arrays can be noticeable. Operating in-place also improves quicksort's locality.  
  
When working with linked lists, neither of these advantages necessarily applies. Because linked list cells are often scattered throughout memory, there is no locality bonus to accessing adjacent linked list cells. Consequently, one of quicksort's huge performance advantages is eaten up. Similarly, the benefits of working in-place no longer apply, since merge sort's linked list algorithm doesn't need any extra auxiliary storage space.  
  
That said, quicksort is still very fast on linked lists. Merge sort just tends to be faster because it more evenly splits the lists in half and does less work per iteration to do a merge than to do the partitioning step.

归纳一下，就是说，如果待排序的元素存储在数组中，那么快速排序相对归并排序就有两个原因更快。一是，可以很快地进行元素的读取(相对于链表，数组的元素是顺序摆放的，而链表的元素是随机摆放的)，数组的partion这步就比链表的partion这步快。二是，归并排序在merge阶段需要辅助数组，需要申请O(N)的空间，申请空间也是需要时间的。而快排不需要额外申请空间。如果待排序的元素存储在链表中，快排的优点就变成了缺点。归并排序于是就速度更优了。