习题 11 部分参考答案

11.1 设要将序列（32, 60, 8, 70, 97, 75, 53, 26, 54, 61）按非递减顺序重新排列，则：

冒泡排序一趟的结果是_{32, 8, 60, 70, 75, 53, 26, 54, 61, 97}_；

插入排序一趟的结果是__{32, 60, 8, 70, 97, 75, 53, 26, 54, 61}_；

二路归并排序一趟的结是__{32, 60, 8, 70, 75, 97, 26, 53, 54, 61}_；

快速排序一趟的结果（以原首元素为枢轴）是_{8, 26, 32, 70, 97, 75, 53, 60, 54, 61}_；

上述算法中稳定的排序算法有__冒泡排序、插入排序、二路归并排序_；

*不稳定的有：选择排序，快速排序，希尔排序，堆排序

(1)冒泡排序

冒泡排序就是把小的元素往前调或者把大的元素往后调。比较是相邻的两个元素比较，交换也发生在这两个元素之间。所以，如果两个元素相等，是不会无聊地把他们俩交换一下的；如果两个相等的元素没有相邻，那么即使通过前面的两两交换把两个相邻起来，这时候也不会交换，所以相同元素的前后顺序并没有改变，所以冒泡排序是一种稳定排序算法。

(2)选择排序

选择排序是给每个位置选择当前元素最小的，比如给第一个位置选择最小的，在剩余元素里面给第二个元素选择第二小的，依次类推，直到第 n-1 个元素，第 n 个元素不用选择了，因为只剩下它一个最大的元素了。那么，在一趟选择，如果当前元素比一个元素小，而该小的元素又出现在一个和当前元素相等的元素后面，那么交换后稳定性就被破坏了。比较拗口，举个例子，序列 5 8 5 2 9， 我们知道第一遍选择第 1 个元素 5 会和 2 交换，那么原序列中 2 个 5 的相对前后顺序就被破坏了，所以选择排序不是一个稳定的排序算法。

(3)插入排序

插入排序是在一个已经有序的小序列的基础上，一次插入一个元素。当然，刚开始这个有序的小序列只有 1 个元素，就是第一个元素。比较是从有序序列的末尾开始，也就是想要插入的元素和已经有序的最大者开始比起，如果比它大则直接插入在其后面，否则一直往前找直到找到它该插入的位置。如果碰见一个和插入元素相等的，那么插入元素把想插入的元素放在相等元素的后面。所以，相等元素的前后顺序没有改变，从原无序序列出去的顺序就是排好序后的顺序，所以插入排序是稳定的。

(4)快速排序

快速排序有两个方向，左边的 i 下标一直往右走，当 a[i] <= a[center_index]，其中 center_index 是中枢元素的数组下标，一般取为数组第 0 个元素。而右边的 j 下标一直往左走，当 a[j] > a[center_index]。如果 i 和 j 都走不动了，i <= j，交换 a[i]和 a[j],重复上面的过程，直到 i>j。 交换 a[j]和 a[center_index]，完成一趟快速排序。在中枢元素和 a[j] 交换的时候，很有

可能把前面的元素的稳定性打乱，比如序列为 5 3 3 4 3 8 9 10 11， 现在中枢元素 5 和 3(第 5 个元素，下标从 1 开始计)交换就会把元素 3 的稳定性打乱，所以快速排序是一个不稳定的排序算法，不稳定发生在中枢元素和 a[j] 交换的时刻。

(5)归并排序

归并排序是把序列递归地分成短序列，递归出口是短序列只有 1 个元素(认为直接有序)或者 2 个序列(1 次比较和交换),然后把各个有序的段序列合并成一个有序的长序列，不断合并直到原序列全部排好序。可以发现，在 1 个或 2 个元素时，1 个元素不会交换，2 个元素如果大小相等也没有人故意交换，这不会破坏稳定性。那么，在短的有序序列合并的过程中，稳定是否受到破坏？没有，合并过程中我们可以保证如果两个当前元素相等时，我们把处在前面的序列的元素保存在结 果序列的前面，这样就保证了稳定性。所以，归并排序也是稳定的排序算法。

(6)基数排序

基数排序是按照低位先排序，然后收集；再按照高位排序，然后再收集；依次类推，直到最高位。有时候有些属性是有优先级顺序的，先按低优先级排序，再按高优 先级排序，最后的次序就是高优先级高的在前，高优先级相同的低优先级高的在前。基数排序基于分别排序，分别收集，所以其是稳定的排序算法。

(7)希尔排序(shell)

希尔排序是按照不同步长对元素进行插入排序，当刚开始元素很无序的时候，步长最大，所以插入排序的元素个数很少，速度很快；当元素基本有序了，步长很小， 插入排序对于有序的序列效率很高。所以，希尔排序的时间复杂度会比 o(n^2)好一些。由于多次插入排序，我们知道一次插入排序是稳定的，不会改变相同元素的相对顺序，但在不同的插入排序过程中，相同的元素可能在各自的插入排序中移动，最后其稳定性就会被打乱，所以 shell 排序是不稳定的。

(8)堆排序

我们知道堆的结构是节点 i 的孩子为 2*i 和 2*i+1 节点，大顶堆要求父节点大于等于其 2 个子节点，小顶堆要求父节点小于等于其 2 个子节点。在一个长为 n 的序列，堆排序的过程是从第 n/2 开始和其子节点共 3 个值选择最大(大顶堆)或者最小(小顶堆),这 3 个元素之间的选择当然不会破坏稳定性。但当为 n /2-1, n/2-2, ...1 这些个父节点选择元素时，就会破坏稳定性。有可能第 n/2 个父节点交换把后面一个元素交换过去了，而第 n/2-1 个父节点把后面一个相同的元素没 有交换，那么这 2 个相同的元素之间的稳定性就被破坏了。所以，堆排序不是稳定的排序算法。

11.2 设有一个待排序的数据序列，其关键字序列如下：{3, 17, 12, 61, 8, 70, 97, 75, 53, 26, 54, 61}，试写出下列排序算法对这个数据序列进行排序的中间及最终结果：

1. 直接插入排序。
2. 希尔排序。
3. 冒泡排序。

4. 快速排序。

5. 选择排序。

6. 归并排序。

解答：

1. 直接插入排序的过程。

数据序列:　　3 17 12 61 8 70 97 75 53 26 54 61

第 1 趟排序后: 3 17 12 61 8 70 97 75 53 26 54 61

第 2 趟排序后: 3 12 17 61 8 70 97 75 53 26 54 61

第 3 趟排序后: 3 12 17 61 8 70 97 75 53 26 54 61

第 4 趟排序后: 3 8 12 17 61 70 97 75 53 26 54 61

第 5 趟排序后: 3 8 12 17 61 70 97 75 53 26 54 61

第 6 趟排序后: 3 8 12 17 61 70 97 75 53 26 54 61

第 7 趟排序后: 3 8 12 17 61 70 75 97 53 26 54 61

第 8 趟排序后: 3 8 12 17 53 61 70 75 97 26 54 61

第 9 趟排序后: 3 8 12 17 26 53 61 70 75 97 54 61

第 10 趟排序后: 3 8 12 17 26 53 54 61 70 75 97 61

第 11 趟排序后: 3 8 12 17 26 53 54 61 61 70 75 97

2. 希尔排序的过程。

数据序列:　　3 17 12 61 8 70 97 75 53 26 54 61

jump=6　第 1 趟排序后: 3 17 12 26 8 61 97 75 53 61 54 70

jump=3　第 2 趟排序后: 3 8 12 26 17 53 61 54 61 97 75 70

jump=1　第 3 趟排序后: 3 8 12 17 26 53 54 61 61 70 75 97

3. 冒泡排序的过程。

数据序列:　　3 17 12 61 8 70 97 75 53 26 54 61

第 1 趟排序后: 3 12 17 8 61 70 75 53 26 54 61 97

第 2 趟排序后: 3 12 8 17 61 70 53 26 54 61 75 97

第 3 趟排序后: 3 8 12 17 61 53 26 54 61 70 75 97

第 4 趟排序后: 3 8 12 17 53 26 54 61 61 70 75 97

第 5 趟排序后: 3 8 12 17 26 53 54 61 61 70 75 97

第 6 趟排序后: 3 8 12 17 26 53 54 61 61 70 75 97

4. 快速排序的过程。

数据序列: 3 17 12 61 8 70 97 75 53 26 54 61

left=0 right=11 Pivot=0 数据序列: 3 17 12 61 8 70 97 75 53 26 54 61

left=1 right=11 Pivot=3 数据序列: 3 8 12 17 61 70 97 75 53 26 54 61

left=1 right=2 Pivot=1 数据序列: 3 8 12 17 61 70 97 75 53 26 54 61

left=4 right=11 Pivot=8 数据序列: 3 8 12 17 53 61 54 26 61 75 97 70

left=4 right=7 Pivot=5 数据序列: 3 8 12 17 26 53 54 61 61 75 97 70

left=6 right=7 Pivot=6 数据序列: 3 8 12 17 26 53 54 61 61 75 97 70

left=9 right=11 Pivot=10 数据序列: 3 8 12 17 26 53 54 61 61 70 75 97

5. 选择排序的过程。

数据序列: 3 17 12 61 8 70 97 75 53 26 54 61

min=0 第 1 趟排序后: 3 17 12 61 8 70 97 75 53 26 54 61

min=4 第 2 趟排序后: 3 8 12 61 17 70 97 75 53 26 54 61

min=2 第 3 趟排序后: 3 8 12 61 17 70 97 75 53 26 54 61

min=4 第 4 趟排序后: 3 8 12 17 61 70 97 75 53 26 54 61

min=9 第 5 趟排序后: 3 8 12 17 26 70 97 75 53 61 54 61

min=8 第 6 趟排序后: 3 8 12 17 26 53 97 75 70 61 54 61

min=10 第 7 趟排序后: 3 8 12 17 26 53 54 75 70 61 97 61

min=9 第 8 趟排序后: 3 8 12 17 26 53 54 61 70 75 97 61

min=11 第 9 趟排序后: 3 8 12 17 26 53 54 61 61 75 97 70

min=11 第 10 趟排序后: 3 8 12 17 26 53 54 61 61 70 97 75

min=11 第 11 趟排序后: 3 8 12 17 26 53 54 61 61 70 75 97

6. 归并排序的过程。

数据序列: 3 17 12 61 8 70 97 75 53 26 54 61

len=1 数据序列: 3 17 12 61 8 70 75 97 26 53 54 61

len=2 数据序列: 3 12 17 61 8 70 75 97 26 53 54 61

len=4 数据序列: 3 8 12 17 61 70 75 97 26 53 54 61

len=8 数据序列: 3 8 12 17 26 53 54 61 61 70 75 97

11.2 说明本章介绍的各个排序算法的特点，并比较它们的的时间复杂度与空间复杂度。

解答：

直接插入排序：在第m趟插入第m个数据元素k时，前m-1个数据元素已组成有序数据序列S_{m-1}，将k 与S_{m-1} 中各数据元素依次进行比较并插入到适当位置，得到新的序列S_m仍是有序的。

希尔排序：先将整个序列分割成若干子序列分别进行直接插入排序，待整个序列基本有序时，再进行全序列直接插入排序，这样可使排序过程加快。希尔排序算法在排序之初，进行比较的是相隔较远的数据元素，使得数据元素移动时能够跨越多个位置；然后逐渐减少被比较数据元素间的距离（缩小增量），直至距离为1 时，各数据元素都已按序排好。

冒泡排序：将相邻的的两个数据元素按关键字进行比较，如果反序，则交换。对于一个待排序的数据序列，经过一趟排序后，最大值数据元素移到最后位置，值较小的数据元素向最终位置移动一位，此过程又称为一趟起泡。如果在一趟排序中，没有发生一次数据交换（起泡），则说明序列已排好序。

快速排序：将长序列分成两个独立的子序列，第一个子序列的元素的关键字均比第二个子序列的元素的关键字小，分别对两个子序列继续进行排序，直到整个序列有序。在待排序的数据序列中任意选择一个元素（如第一个元素）作为基准值pivot，由序列的两端交替地向中间进行比较、交换，使得所有比pivot 小的元素都处于序列的左端，所有比pivot 大的元素都处于序列的右端，这样序列就被划分成两个小序列。再对两个子序列分别进行同样的操作，直到子序列的长度为1。

直接选择排序：对于有n 个元素的待排序数据序列，第1 趟排序，比较n 个元素，找到关键字最小的元素items[min]，将其交换到序列的首位置items[0]；第2 趟排序，在余下的n-1 个元素中选取最小的元素，交换到序列的位置1；这样经过n-1 趟排序，完成n 个元素的排序。

堆排序：在每次选择最小或最大值时，利用以前的比较结果以提高排序的速度。

归并排序：

1) 将待排序序列看成是n 个长度为1 的已排序子序列。
2) 依次将两个子序列合并成一个大的有序序列。
3) 重复第2 步，合并更大的有序子序列，直到合并成一个序列，排序完成。

各种排序算法的性能比较

	时间复杂度			空间复杂度	稳定性
	平均情况	最坏情况	最好情况		
直接插入	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	稳定的
希尔排序	$O(n(\log_2 n)^2)$			$O(1)$	不稳定
直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	$O(1)$	不稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定的
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	$O(\log_2 n)$	不稳定
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定的

11.3 排序算法的稳定性的含义是什么？说明本章介绍的各个排序算法的稳定性。

解答：

在数据序列中，如果有两个数据元素r_i 和r_j，它们的关键字k_i 等于k_j，且在未排序时，

r_i 位于r_j 之前。如果排序后，元素r_i 仍在r_j 之前，则称这样的排序算法是稳定的（stable），否则是不稳定的排序算法。各个排序算法的稳定性见上表。

11.4 排序的关键字不同，排序的结果也不一样。说明C#程序中指定排序关键字的一些方法。

解答：

C#程序中，关键字是通过在被处理的数据类（型）定义中实现由 IComparable 接口定义的 CompareTo()方法指定的，在该方法中指定一个关键字承接比较功能。调用 Array.Sort 方法即可完成由这种类型的数据元素组成的序列、按这个指定的关键字的排序。如果要按其他的关键字比较，则需向 Array.Sort 方法提供一个实现 IComparer 接口的“比较器”对象，由它（该“比较器”）定义比较的规则（主要是指定比较所需的关键字）。

11.6 分析用冒泡排序对数据序列items={70, 30, 12, 61, 80, 20, 97, 46}进行升序排序所需的比较操作的总次数。

解答：

第 1 趟排序后: 30 12 61 70 20 80 46 97
第 2 趟排序后: 12 30 61 20 70 46 80 97
第 3 趟排序后: 12 30 20 61 46 70 80 97
第 4 趟排序后: 12 20 30 46 61 70 80 97
第 5 趟排序后: 12 20 30 46 61 70 80 97
排序后数据序列: 12 20 30 46 61 70 80 97
可知比较操作的总次数为 sigma(n-m), m=1,…,5, 即 7+6+5+4+3=25 次。

11.7 分析快速排序在最好情况和最坏情况下的时间复杂度。

解答：

快速排序的执行时间与序列的初始排列及基准值的选取有关。最坏情况是，当序列已排序时，例如{1, 2, 3, 4, 5, 6, 7, 8}，如果选取序列的第一个值作为基准值，那么分成的两个子序列将分别是{1}和{2, 3, 4, 5, 6, 7, 8}，而且它们仍然是已排序的。这样必须经过 n-1=7 趟才能完成最终的排序。在这种情况下，其时间复杂度为 $O(n^2)$，排序速度已退化，比冒泡法还慢。

快速排序的最好情况是，每趟排序将序列分成两个长度相同的子序列，此时时间复杂度为 $O(n\log_2 n)$