练习

2.5.1 在下面这段 String 类型的 compareTo() 方法的实现中，第三行对提高运行效率有何帮助？

```
public int compareTo(String that)
{
   if (this == that) return 0;  // 这一行
   int n = Math.min(this.length(), that.length());
   for (int i = 0; i < n; i++)
   {
      if      (this.charAt(i) < that.charAt(i)) return -1;
      else if (this.charAt(i) > that.charAt(i)) return +1;
   }
   return this.length() - that.length();
}
```

2.5.2 编写一段程序，从标准输入读入一列单词并打印出其中所有由两个单词组成的组合词。例如，如果输入的单词为 after、thought 和 afterthought，那么 afterthought 就是一个组合词。

2.5.3 找出下面这段账户余额 Balance 类的实现代码的错误。为什么 compareTo() 方法对 Comparable 接口的实现有缺陷？

```
public class Balance implements Comparable<Balance>
{
   ...
   private double amount;
   public int compareTo(Balance that)
   {
      if (this.amount < that.amount - 0.005) return
-1;
      if (this.amount > that.amount + 0.005) return
+1;
      return 0;
   }
   ...
}
```

说明如何修正这个问题。

2.5.4 实现一个方法 String[] dedup(String[] a)，返回一个有序的 a[]，并删去其中的重复元素。

2.5.5 说明为何选择排序是不稳定的。

2.5.6 用递归实现 select()。

2.5.7 用 select() 找出 N 个元素中的最小值平均大约需要多少次比较？

2.5.8 编写一段程序 Frequency，从标准输入读取一列字符串并按照字符串出现频率由高到低的顺序打印出每个字符串及其出现次数。

2.5.9 为将右侧所示的数据排序编写一个新的数据类型。

2.5.10 创建一个数据类型 Version 来表示软件的版本，例如 115.1.1、115.10.1、115.10.2。为它实现 Comparable 接口，其中 115.1.1 的版本低于 115.10.1。

输入DJIA每天的成交量

```
 1-Oct-28      3500000
 2-Oct-28      3850000
 3-Oct-28      4060000
 4-Oct-28      4330000
 5-Oct-28      4360000
...
30-Dec-99    554680000
31-Dec-99    374049984
 3-Jan-00    931800000
 4-Jan-00   1009000000
 5-Jan-00   1085500032
 ...
```

输出

```
19-Aug-40 130000
26-Aug-40 160000
24-Jul-40 200000
10-Aug-42 210000
23-Jun-42 210000
...
23-Jul-02 2441019904
17-Jul-02 2566500096
15-Jul-02 2574799872
19-Jul-02 2654099968
24-Jul-02 2775559936
```

2.5.11 描述排序结果的一种方法是创建一个保存 0 到 a.length–1 的排列 p[]，使得 p[i] 的值为 a[i] 元素的最终位置。用这种方法描述插入排序、选择排序、希尔排序、归并排序、快速排序和堆排序对一个含有 7 个相同元素的数组的排序结果。 353 ⌇ 354

提高题

2.5.12 调度。编写一段程序 SPT.java，从标准输入中读取任务的名称和所需的运行时间，用 2.5.4.3 节所述的最短处理时间优先的原则打印出一份调度计划，使得任务完成的平均时间最小。

2.5.13 负载均衡。编写一段程序 LPT.java，接受一个整数 M 作为命令行参数，从标准输入中读取任务的名称和所需的运行时间，用 2.5.4.3 节所述的最长处理时间优先原则打印出一份调度计划，将所有任务分配给 M 个处理器并使得所有任务完成所需的总时间最少。

2.5.14 逆域名排序。为域名编写一个数据类型 Domain 并为它实现一个 compareTo() 方法，使之能够按照逆向的域名排序。例如，域名 cs.princeton.edu 的逆是 edu.princeton.cs。这在网络日志处理时很有用。提示：使用 s.split("\\.") 将域名用点分为若干部分。编写一个 Domain 的用例，从标准输入读取域名并将它们按照逆域名有序地打印出来。

2.5.15 垃圾邮件大战。在非法垃圾邮件之战的伊始，你有一大串来自各个域名（也就是电子邮件地址中 @ 符号后面的部分）的电子邮件地址。为了更好地伪造回信地址，你应该总是从相同的域中向目标用户发送邮件。例如，从 wayne@cs.princeton.edu 向 rs@cs.princeton.edu 发送垃圾邮件就很不错。你会如何处理这份电子邮件列表来高效地完成这个任务呢？

2.5.16 公正的选举。为了避免对名字排在字母表靠后的候选人的偏见，加州在 2003 年的州长选举中将所有候选人按照以下字母顺序排列：

R W Q O J M V A H B S G Z X N T C I E K U P D Y F L

创建一个遵守这种顺序的数据类型并编写一个用例 California，在它的静态方法 main() 中将字符串按照这种方式排序。假设所有字符串全部都是大写的。

2.5.17 检测稳定性。扩展练习 2.1.16 中的 check() 方法，对指定数组调用 sort()，如果排序结果是稳定的则返回 true，否则返回 false。不要假设 sort() 只会使用 exch() 移动数据。 355

2.5.18 强制稳定。编写一段能够将任意排序方法变得稳定的封装代码，创建一种新的数据类型作为键，将键的原始索引保存在其中，并在调用 sort() 之后再根据保存的索引恢复键的原始顺序。

2.5.19 Kendall tau 距离。编写一段程序 KendallTau.java，在线性对数时间内计算两组排列之间的 Kendall tau 距离。

2.5.20 空闲时间。假设有一台计算机能够并行处理 N 个任务。编写一段程序并给定一系列任务的起始时间和结束时间，找出这台机器最长的空闲时间和最长的繁忙时间。

2.5.21 多维排序。编写一个 Vector 数据类型并将 d 维整型向量排序。排序方法是先按照一维数字排序，一维数字相同的向量则按照二维数字排序，再相同的向量则按照三维数字排序，如此这般。

2.5.22 股票交易。投资者对一只股票的买卖交易都发布在电子交易市场中。他们会指定最高买入价和最低卖出价，以及在该价位买卖的笔数。编写一段程序，用优先队列来匹配买家和卖家并用模拟数据进行测试。可以使用两个优先队列，一个用于买家一个用于卖家，当一方的报价能够和另一方的一份或多份报价匹配时就进行交易。

2.5.23 选择的取样：实验使用取样来改进 select() 函数的想法。提示：使用中位数可能并不总是有效。

2.5.24 稳定的优先队列。实现一个稳定的优先队列（将重复的元素按照它们被插入的顺序返回）

2.5.25 平面上的点。为表 1.2.3 的 Point2D 类型编写三个静态的比较器，一个按照 x 坐标比较，一个按照 y 坐标比较，一个按照点到原点的距离进行比较。编写两个非静态的比较器，一个按照两点到第三点的距离比较，一个按照两点相对于第三点的幅角比较。

356 2.5.26 简单多边形。给定平面上的 N 个点，用它们画出一个多边形。提示：找到 y 坐标最小的点 p，在有多个最小 y 坐标的点时取 x 坐标最小者，然后将其他点按照以 p 为原点的幅角大小的顺序依次连接起来。

2.5.27 平行数组的排序。在将平行数组排序时，可以将索引排序并返回一个 index[] 数组。为 Insertion 添加一个 indirectSort() 方法，接受一个 Comparable 的对象数组 a[] 作为参数，但它不会将 a[] 中的元素重新排列，而是返回一个整形数组 index[] 使得 a[index[0]] 到 a[index[N-1]] 正好是升序的。

2.5.28 按文件名排序。编写一个 FileSorter 程序，从命令行接受一个目录名并打印出按照文件名排序后的所有文件。提示：使用 File 数据类型。

2.5.29 按大小和最后修改日期将文件排序。为 File 数据类型编写比较器，使之能够将文件按照大小、文件名或最后修改日期将文件升序或者降序排列。在程序 LS 中使用这些比较器，它接受一个命令行参数并根据指定的顺序列出目录的内容。例如，"-t" 指按照时间戳排序。支持多个选项以消除排序位次相同者，同时必须确保排序的稳定性。

2.5.30 Boerner 定理。真假判断：如果你先将一个矩阵的每一列排序，再将矩阵的每一行排序，所有的
357 列仍然是有序的。证明你的结论。

实验题

2.5.31 重复元素。编写一段程序，接受命令行参数 M、N 和 T，然后使用正文中的代码进行 T 遍实验：生成 N 个 0 到 M-1 间的 int 值并计算重复值的个数。令 T=10，N=10^3、10^4、10^5 和 10^6 以及 M=N/2、N 和 2N。根据概率论，重复值的个数应该约为 $(1-e^{-\alpha})$，其中 α=N/M。打印一张表格来确认你的实验验证了这个公式。

2.5.32 8 字谜题。8 字谜题是 S. Ioyd 于 19 世纪 70 年代发明的一个游戏。游戏需要一个三乘三的九宫格，其中八格中填上了 1 到 8 这 8 个数字，一格空着。你的目标就是将所有的格子排序。可以将一个格子向上下或者左右移动（但不能是对角线方向）到空白的格子中。编写一个程序用 A* 算法解决这个问题。先用到达九宫格的当前位置所需的步数加上错位的格子数量作为优先级函数（注意，步数至少大于等于错位的格子数）。尝试用其他函数代替错位的格子数量，比如每个格子距离它的正确位置的曼哈顿距离，或是这些距离的平方之和。

2.5.33 随机交易。开发一个接受参数 N 的生成器，根据你能想到的任意假设条件生成 N 个随机的 Transaction 对象（请见练习 2.1.21 和练习 2.1.22）。对于 N=10^3、10^4、10^5 和 10^6，比较用希尔排序、
358 归并排序、快速排序和堆排序将 N 个交易排序的性能。