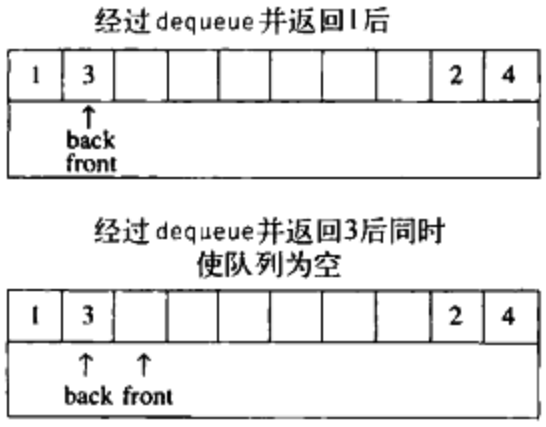

有些程序设计员使用不同的方法表示队列的队头和队尾。例如，有人不使用一项来记录大小，因为他们依赖于当队列为空(back = front - 1)时的基准情形。队列的大小通过比较 back 和 front 隐式地算出。这是一种非常隐秘的方法，因为存在某些特殊的情形，因此，如果你想修改用这种方法编写的程序，那就要特别地小心。如果 currentSize 不作为明确的数据域被保留，那么当存在 theArray.length-1 个元素时队列就满了，因为只有 theArray.length 个不同的大小可被区分，而 0 是其中的一个。可以采用任意一种你喜欢的风格，但要确保你的所有例程都是一致的。由于实现方法有多种选择，因此如果不使用 currentSize 域，那就很可能有必要进行一些注释，否则会在一个程序中使用两种选择。

在保证 enqueue 的次数不会大于队列容量的应用中，使用回绕是没有必要的。像栈一样，除非主调例程肯定队列非空，否则 dequeue 很少执行。因此对这种操作，只要不是关键的代码，错误检测常常被跳过。一般说来这并不是无可非议的，因为这样可能得到的时间节省量是极小的。

3.7.3 队列的应用

有许多使用队列给出高效运行时间的算法。它们当中有些可以在图论中找到，我们将在第 9 章讨论它们。这里，先给出某些应用队列的简单例子。

当作业送交给一台行式打印机的时候，它们就以到达的顺序被排列起来。因此，被送往行式打印机的作业基本上被放到一个队列中。[⊖]

事实上每一个实际生活中的排队都(应该)是一个队列。例如，在一些售票口排列的队伍都是队列，因为服务的顺序是先到先买票。

另一个例子是关于计算机网络的。有多种 PC 机的网络设置，其中磁盘是放在一台叫做**文件服务器**(file server)的机器上的。使用其他计算机的用户是按照先到先使用的原则访问文件的，因此其数据结构是一个队列。

进一步的例子如下：

- 当所有的接线员忙不开的时候，对大公司的呼叫一般都被放到一个队列中。
- 在大型的大学里，如果所有的终端都被占用，由于资源有限，学生们必须在一个等待表上签字登记。在终端上呆得时间最长的学生将首先被强制离开，而等待时间最长的学生则将是下一个被允许使用终端的用户。

称为**排队论**(queueing theory)的整个数学分支处理用概率的方法计算用户预计要排队等待多长时间才会得到服务、等待服务的队伍能够排多长、以及其他一些诸如此类的问题。问题的答案依赖于用户到达排队的经常程度以及一旦用户得到服务时处理服务花费的时间。这两个参数作为概率分布函数给出。在一些简单的情况下，答案可以解析地算出。一种简单情况的例子是一条电话线有一个接线员。如果接线员忙，打来的电话就被放到一个等待队列中(这还与某个容许

⊖ 我们说基本上是因为作业可以被取消。这等于从队列的中间进行一次删除，它违反了队列的严格定义。