请看表4-4的第五行。锁处于上锁状态。使用发卡并顺时针转动不会引起锁的状态改变，也不会给锁造成损坏。执行测试时一切没有改变，则这个正面测试用例通过。这是正面测试的“负面测试条件”的一个例子。

负面测试展示当输入非预期输入时产品没有失败。负面测试的目的是尝试使系统垮掉。负面测试使用产品没有设计和编码的场景。换句话说，输入值可能没有在产品的需求规格说明中描述。就规格说明来说，这些测试条件可以归纳为产品的未知条件。但是对于最终用户，会遇到多种产品需要考虑的场景。了解可能在最终用户层面上出现的负面情况，以便测试并采取预防措施，这一点对于测试人员来说甚至更重要。负面测试检验产品是否在应该提供错误信息时而没有提供，或不应该提供错误信息时却提供了。

正面测试用于验证已知测试条件，负面测试用于通过未知条件把产品搞垮。

表4-5给出了锁和钥匙例子的一些负面测试用例。

表4-5 负面测试用例

序号	输入1	输入2	当前状态	预期状态
1	某个其他锁的钥匙	顺时针转动	上锁	上锁
2	某个其他锁的钥匙	逆时针转动	开锁	开锁
3	铁丝	逆时针转动	开锁	开锁
4	用石头打击		上锁	上锁

在表4-5中，与我们已经看到的正面测试不同，这里没有需求标识。这是因为负面测试关注的是需求规格说明之外的测试条件。由于所有测试条件都在需求规格说明之外，因此不能按正面测试条件和负面测试条件分类。有人认为所有这样的条件都是负面测试条件，从技术角度看这种看法是正确的。

正面测试和负面测试的差别在于它们的覆盖率计算方法。对于正面测试，如果覆盖了所有形成文档的需求和条件，那么其覆盖率就认为是百分之百。如果规格说明很清楚，那么可以得到清晰的覆盖率指标。形成对比的是，负面测试是没有穷尽的，百分之百的负面测试覆盖率是不现实的。负面测试需要测试人员具有高度的创造性来覆盖尽可能多的“未知”条件，以避免产品在客户场地出现故障。

4.4.3 边界值分析

上一节已经提到过，条件和边界是软件产品中的两个主要缺陷源。本节将详细讨论条件问题。大多数软件产品中的缺陷都与条件和边界有关。所谓条件，是指基于各种变量取值需要采取一定行动的情况。所谓边界，是指各种变量值的“极限”。

本节要研究边界值分析（BVA），这是能够有效捕获出现在边界处的缺陷的一种测试方法。边界值分析利用并扩展了缺陷更容易出现在边界处的概念。

为了说明在边界处出现错误的概念，这里举一个能够为客户提供大宗购买折扣的记账系统。

我们很多人都熟悉购物时的大宗购买折扣概念，例如买1件要付1.59美元，但是买3件只需付4美元。大宗购买已经成为购物者的获利原则。从销售商角度看，大宗卖出也是更合算的，因为需要支出较少的存储成本并保持更好的现金流。假设有一家出售各种商品的

商店，它为购买不同数量商品的客户报出不同的价格，也就是说按购买量的不同“分段”计价。

购买数量	单价（美元）
头10件（即从第1件到第10件）	5.00
第二个10件（即从第11件到第20件）	4.75
第三个10件（即从第21件到第30件）	4.50
超过30件	4.00

从上表可以清楚地看出，买5件需要支付5×5=25美元。如果买11件，第一个10件需要支付10×5=50美元，第11件需要支付4.75美元。类似地，如果买15件，需要支付10×5+5×4.75=73.75美元。

从测试角度看，对于这个例子什么类型的数据最有可能暴露出程序的更多缺陷呢？人们通过总结发现，大多数缺陷出现在边界数据附近，例如购买9、10、11、19、20、21、29、30、31件和类似数量的商品时。虽然出现这种现象的原因还没有完全弄清楚，不过还是可以给出以下一些可能的原因：

- 程序员使用合适比较操作符的习惯，例如在进行比较时使用<=操作符，还是<操作符。
- 由于实现循环和条件检查有多种方式而产生的困惑。例如，在像C这样的程序设计语言中，有`for`循环、`while`循环和`repeat`循环。这些循环有不同的终止条件，在确定要使用的操作符时会产生一定程度的困惑，因此会在边界条件附近引入缺陷。
- 可能没有清楚地理解需求本身，特别是对边界附近需求的理解，因此使得即使是正确编码的程序也不能进行正确地处理。

表4-6给出在上面的例子中，应该执行的测试和预期的变量输出值（所购商品的总价）。表4-6只包含了正面测试用例，没有包括诸如非数字输入的负面测试用例。画圈的行是边界值，这些行与其他部分相比更有可能发现缺陷。

表4-6　大宗购买折扣例子的边界值

要测试的输入值	选择测试的理由	预期输出（美元）
1	第一个计价段的开始	5.00
5	第一个计价段中的值，没有考虑边界	25.00
9	正好低于第二个计价段，或正好在第一个计价段的末尾	45.00
10	第二个计价段的极限	50.00
11	正好高于第一个计价段，正好进入第二个计价段	54.75
16	第二个计价段中的值，没有考虑边界	28.50
19	正好低于第三个计价段，或正好在第二个计价段的末尾	92.75
20	第三个计价段的极限	97.50
21	正好高于第二个计价段，正好进入第三个计价段	102.00
27	第三个计价段中的值，没有考虑边界	129.00
29	正好低于第四个计价段，或正好在第三个计价段的末尾	138.00
30	第四个计价段的极限	142.50
31	正好高于第四个计价段	146.50
50	高出第四个计价段低限很多	182.50

边界值测试对于发现缺陷极为重要的另一个例子是特定资源、变量或数据结构的内部极限值。举一个在共享内存区缓存最近使用过的数据块的数据库管理系统（或文件系统）的例子。通常这种缓存区受到用户在启动系统时指定的参数限制。假设数据库在启动时指定缓存最近50个数据库缓存区，那么当这些缓存区满而第51个缓存区需要缓存时，第一个缓存区也就是最先使用的缓存区，就需要转存到次级存储器上。可以看出，操作插入新的缓存区和释放第一个缓存区，都出现在“边界”上。

如果输入（或输出）数据有可以明确区分的边界或范围，那么边界值分析对于生成测试用例是非常有用的。

如图4-2所示，有以下四种情况需要测试。第一，当缓存区完全空着（看起来像一个在逻辑上矛盾的句子）；第二，当插入缓存区，缓存区还有空余时；第三，当插入最后一块缓存区时；第四，当缓存区已满再插入缓存区时。后两种测试比前两种更有可能发现缺陷。

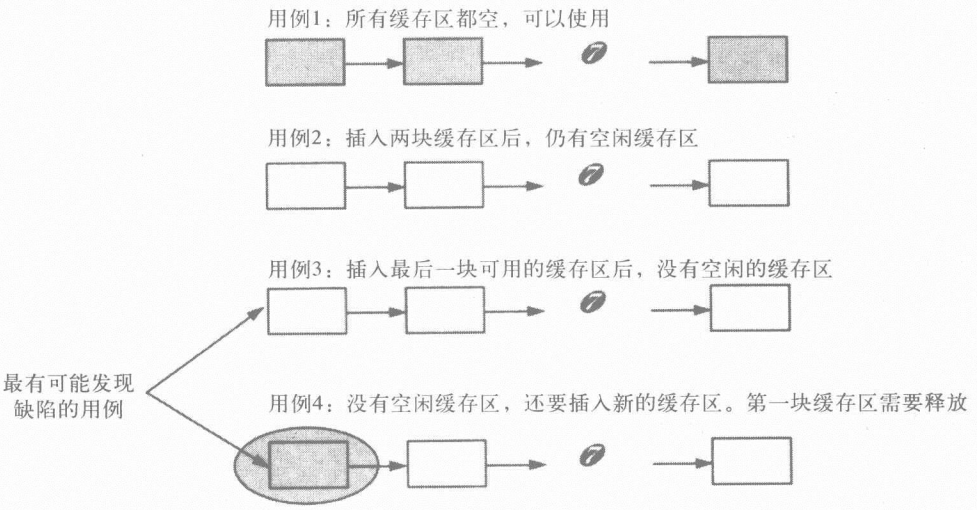

图4-2 缓存区管理的各种测试用例

总结一下边界值测试：

- 检查数据值对计算有影响的级差或不连续点，不连续点就是边界值，需要彻底测试。
- 检查内部极限，例如资源极限（如上面给出的缓存例子）。产品处于这类极限的行为也应该是边界值测试的内容。
- 包含在边界值测试内容中的还有在文档中已说明的对硬件资源的限制。例如，如果文档说明产品将运行在4MB以上的RAM，那么一定要设计测试最低RAM（这里是4MB）的测试用例。
- 前面给出的例子讨论的都是对于输入数据的边界条件，对于输出值也要进行同样的边界值分析。

针对黑盒测试讨论的边界值分析也适用于白盒测试。像数组、堆栈和队列这样的内部数据结构也需要检查边界或极限条件。如果内部使用了链表结构，那么就应该彻底测试链表开始和结尾的行为。

边界值和决策表有助于确定最有可能发现缺陷的测试用例。这些概念的汇总就是本章中下面要讨论的*等价类*概念。