黑盒测试设计

1　概述

本章介绍黑盒测试的概念和进行黑盒测试的目的与意义，及关于等价类划分、边界值分析、因果图法、判定表法、正交试验法、功能图法等测试用例设计方法的原理与实现，并从测试设计说明、测试用例说明、测试程序说明三个方面介绍如何编写测试用例，最后结合一个 ATM 的例子体现如何设计测试用例。黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。

黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。黑盒测试是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。很明显，如果外部特性本身设计有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。

1)　黑盒测试主要测试的错误类型

黑盒测试法注重于测试软件的功能需求，主要试图发现下列几类错误。

- □ 功能不正确或遗漏
- □ 界面错误
- □ 输入和输出错误
- □ 数据库访问错误
- □ 性能错误
- □ 初始化和终止错误等

2)　对黑盒测试人员的要求

黑盒测试只关心软件的外部功能和界面表现，不接触代码，为了保证测试工作顺利进行，在合理的时间内完成测试，发现软件系统的缺陷，掌握测试用例的书写，保证结果的可靠性，在对黑盒测试人员的选择和要求上也要符合一定的标准：

□ 掌握测试思想和常规的流程。

□ 了解产品的需求和功能

□ 掌握测试用例的书写

□ 有一定的软件开发和测试经验

2 测试用例的编写

2.1 测试用例定义

所谓的测试用例设计就是将软件测试的行为活动，作一个科学化的组织归纳。软件测试是有组织性、步骤性和计划性的，而设计软件测试用例的目的，就是为了能将软件测试的行为转换为可管理的模式。软件测试是软件质量管理中最实际的行动，同时也是耗时最多的一项。基于时间因素的考虑，软件测试行为必须能够加以量化，才能进一步让管理阶层掌握所需要的测试过程，而测试用例就是将测试行为具体量化的方法之一。

简单地说，测试用例就是设计一个情况，软件程序在这种情况下，必须能够正常运行并且达到程序所设计的执行结果。如果程序在这种情况下不能正常运行，而且这种问题会重复发生，那就表示软件程序人员已经测出软件有缺陷，这时候就必须将这个问题标示出来，并且输入到问题跟踪系统内，通知软件开发人员。软件开发人员接获通知后，将这个问题修改完成于下一个测试版本内，软件测试工程师取得新的测试版本后，必须利用同一个用例来测试这个问题，确保该问题已修改完成。

因为我们不可能进行穷举测试，为了节省时间和资源、提高测试效率，必须要从数量极大的可用测试数据中精心挑选出具有代表性或特殊性的测试数据来进行测试。

使用测试用例的好处主要体现在以下几个方面。

① 在开始实施测试之前设计好测试用例，可以避免盲目测试并提高测试效率。

② 测试用例的使用令软件测试的实施重点突出、目的明确。

③ 在软件版本更新后只需修正少部分的测试用例便可展开测试工作，降低工作强

度，缩短项目周期。

④ 功能模块的通用化和复用化使软件易于开发，而测试用例的通用化和复用化则会使软件测试易于开展，并随着测试用例的不断精化其效率也不断攀升。

具体的黑盒测试用例设计方法包括等价类划分法、边界值分析法、错误推测法、因果图法、判定表驱动法、正交试验设计法、功能图法等。应该说，这些方法是比较实用的，但采用什么方法，在使用时自然要针对开发项目的特点对方法加以适当的选择。下面我们讨论几种常用的方法。

2.2 测试设计说明

大家知道，项目整体测试计划的级别非常高。它虽然把软件拆分为具体特性和可测试项，并将其分派到每个测试员头上，但是它并没有指明如何对这些特性进行测试，可能仅仅对使用自动化测试还是黑盒测试或者白盒测试有一些提示，但是并不会涉及如何使用以及在哪里使用这些工具。

为了更好地进行测试，我们需要为单个软件特性定义具体的测试方法，这就是测试设计说明。ANSI/IEEE 829 中对测试设计说明的解释是：测试设计说明就是在测试计划中提炼测试方法，要明确指出设计包含的特性以及相关的测试用例和测试程序，并指定判断特性通过/失败的规则。

测试设计说明的目的是组织和描述针对具体特性需要进行的测试。然而，它并不给出具体的测试用例或者执行测试的步骤。以下内容来自于 ANSI/IEEE 829 标准，应该作为测试设计说明的部分内容。

- □ 标识符：用于引用和定位测试设计说明的惟一标识符。该说明还应该引用整个测试计划，还应该包含任何其他计划或者说明的引用。
- □ 要测试的特性：对测试设计说明所包含的软件特性的描述。例如“计算器程序的加法功能”、“写字板程序中的字体大小选择和显示”、“QuickTime 软件的视频卡配置测试”。该部分还将明确指出要间接测试的特性，它通常作为主特性的辅助特性。例如，文件打开对话框的用户界面虽然不在测试设计说明中重点指出，但是在测试读写功能的过程中要间接测试。
- □ 方法：描述测试的通用方法。如果方法在测试计划中列出，就应该在此详细描述要使用的技术，并给出如何验证测试结果的方法。例如，我们这样描述一种方法，开发一种测试工具，顺序读写不同大小的数据文件，数据文件的数目和大小及包含的内容由程序员提供的示例来确定。用文件比较工具比较输出的文件和源文件，如果相同，则认为通过；如果不同，则认为失败。
- □ 测试用例信息：用于描述所引用的测试用例的相关信息。应该列出所选的等价

区间，给出测试用例的引用信息以及用于执行测试用例的测试程序说明。例如：“检查最大值　测试用例 ID#15326”、“检查最小值　测试用例 ID#15327”，在这部分不定义实际测试用例。

□ 通过/失败规则：描述用什么规则来判定某项特性的测试结果是通过还是失败。这种描述有可能非常简单和明确，例如“通过是指当执行全部测试用例时没有发现软件缺陷”。也有可能不是非常明确，例如“失败是指 10% 以上的测试用例没有通过”。

2.3　测试用例说明

如何记录和记载创建的测试用例？如果你已经开始进行一些软件测试了，就可能采用过一些用例描述格式。本节讲解编写测试用例的有关内容，指出将要考虑的重点。

ANSI/IEEE 829 标准称测试用例说明为编写用于输入输出的实际数值和预期结果，同时还明确指出，使用具体测试用例产生的测试程序的限制。一个测试用例的编写可参考表 5-14。

表 5-14　测 试 用 例

编号：

编制人		审定人		时间	
软件名称			编号/版本		
测试用例					
用例编号					

参考信息（参考的文档及章节号或功能项）：

输入说明（列出选用的输入项，覆盖正常、异常情况）：

输出说明（逐条与输入项对应，列出预期输出）：

环境要求（测试要求的软、硬件、网络要求）：

特殊规程要求：

用例间的依赖关系：

用例产生的测试程序限制：

测试用例应该解释要向软件发送什么值或者条件，以及预期结果。一个测试用例说明可以由多个测试用例说明来引用，也可以引用多个测试程序。ANSI/IEEE 829 标准还列出了一些应该包含在内的重要信息，如下所示。

- □ 标识符：由测试设计过程说明和测试程序说明引用的惟一标识符。
- □ 测试项：描述被测试的详细特性、代码模块等，应该比测试设计说明中所列的特性更加具体。如果测试设计说明提到“计算器程序的加法功能”，那么测试用例说明就会相应地提到“加法运算的上限溢出处理”。它还要指出引用的产品说明书或者测试用例所依据的其他设计文档。
- □ 输入说明：该说明列举执行测试用例的所有输入内容或者条件。如果测试计算器程序，输入说明可能简单到“1+1”。如果测试蜂窝电话交换软件，输入说明可能是成百上千种输入条件。如果测试基于文件的产品，输入说明可能是文件名和内容的描述。
- □ 输出说明：描述进行测试用例预期的结果。例如，1+1 等于 2 吗？在蜂窝软件中上千个输出变量设置正确吗？读取文件的全部内容和预想的一样吗？
- □ 环境要求：是指执行测试用例必要的硬件、软件、测试工具、人员等。
- □ 特殊要求：描述执行测试必须的特殊要求。测试写字板程序也许不需要任何特殊条件，但是测试一些特殊的软件（如核电站软件）就有特殊要求。
- □ 用例之间的依赖性：如果一个测试用例依赖于其他用例，或者受其他用例的影响，就应该在此注明。

如果按照这里推荐的文档格式，对于每一个测试用例至少都要写上一页的描述文字，数千个测试用例可能要形成几千页文档。所以我们经常把ANSI/IEEE 829 标准当作规范而不是标准使用（除非必须这样做，许多政府项目和某些行业要求按照此规格编写测试用例，但是在大多数情况下可以采用简便方法）。

采用简便方法并不是说放弃或者忽视重要的信息，而是意在找出一个更有效的方法

对这些信息进行精简，例如，没有必要刻意要求不能用书面段落形式表述测试用例。如表5-15给出了一个打印机兼容性简单列表的例子。

表5-15 打印机兼容性简单列表

测试用例序列号	型 号	模 式	黑 白	选 项
WP0001	Canon	BJC-7000	黑白	文字
WP0002	Canon	BJC-7000	黑白	超级照片
WP0003	Canon	BJC-7000	黑白	自动
WP0004	Canon	BJC-7000	黑白	草稿
WP0005	Canon	BJC-7000	彩色	文字
WP0006	Canon	BJC-7000	彩色	超级照片
WP0007	Canon	BJC-7000	彩色	自动
WP0008	Canon	BJC-7000	彩色	草稿
WP0009	HP	LaserJet Ⅳ	高	
WP0010	HP	LaserJet Ⅳ	中	
WP0011	HP	LaserJet Ⅳ	低	

表中的每一行是一个测试用例，有自己的标识符。伴随测试用例的所有其他信息，例如测试项、输入说明、输出说明、环境要求、特殊要求和依赖性等对所有这些用例都必须有，可以一并编写，附加到表格中。审查测试用例的人可以快速看完测试用例信息，然后审查表格，检查其范围。

表述测试用例的其他选择有大纲、状态表或数据流程图等方式。

2.4 测试程序说明

编写完测试设计和测试用例之后，就要说明执行测试用例的程序。什么是测试程序呢？ANSI/IEEE 829标准把测试程序定义为“明确指出为实现相关测试设计而执行具体测试用例和操作软件系统的全部步骤”。

测试程序，有时也叫“测试脚本说明”，详细定义了执行测试用例的每一步操作。以下是需要定义的内容。

□ 标识符：用来把测试程序与相关测试用例和测试设计相联系的惟一标识。

□ 目的：本程序描述的目的以及将要执行的测试用例的引用信息。

□ 特殊要求：执行测试所需的其他程序、特殊测试技术或者特殊设备。

□ 程序步骤：执行测试用例的详细描述。它包含以下内容。

① 日志：指出用什么方法记录测试结果和现象。

② 设置：说明如何准备测试。

③ 启动：说明启动测试的步骤。

④ 程序：描述运行测试的步骤。

⑤ 衡量标准：描述如何判断结果。

⑥ 关闭：描述因意外原因而推迟测试的步骤。

⑦ 终止：描述正常停止测试的步骤。

⑧ 重置：说明如何把环境恢复到测试前的状态。

⑨ 偶然事件：说明如何处理计划之外的情况。

如果我们把测试程序只理解成“尝试执行所有的测试用例并报告发现的问题”是不够的。这虽然简单、容易，但是无法告诉新加入的测试员如何进行测试，不能重复而且无法证明哪些步骤执行了。使用详细的程序说明，则把要测试什么、如何测试等问题都表述得一目了然。如图 5-12 所示是“Windows 计算器”的测试程序说明的例子片断。

标识号：计算器。

目的：本程序说明描述执行加法测试用例的步骤。

特殊要求：本次测试不需要特殊的硬件和软件。

程序步骤：

日志：测试员按测试要求记录程序执行过程，所有必须填写的项都必须填写，包括问题的记录。

设置：测试者必须安装 Windows 98 的干净副本，使用测试工具 Tool-A 和 Tool-B 等。

启动：启动 Windows 98，单击开始按钮，选择程序，选择附件、选择计算器。

程序：用键盘输入每个测试用例并比较结果。

衡量标准：□□

图 5-12　测试程序说明片断

2 黑盒测试方法

初涉软件测试者可能认为拿到软件后就可以立即进行测试，并希望马上找出软件的所有缺陷，这种想法就如同没有受过工程训练的开发工程师急于去编写代码一样。软件测试也是一个工程，也需要按照工程的角度去认识软件测试，在具体的测试实施之前，我们需要明白我们测试什么，怎么测试等，也就是说通过制定测试用例指导测试的实施。

2.2 等价类划分法

等价类划分是一种典型的黑盒测试方法，用这一方法设计测试用例完全不考虑程序的内部结构，只根据对程序的要求和说明，即需求规格说明书。我们必须仔细分析和推敲说明书的各项需求，特别是功能需求。把说明中对输入的要求和输出的要求区别开来并加以分解。

由于穷举测试工作量太大，以至于无法实际完成，促使我们在大量的可能数据中选取其中的一部分作为测试用例。例如，在不了解等价分配技术的前提下，我们做计算器程序的加法测试时，测试了 1+1，1+2，1+3 和 1+4 之后，还有必要测试 1+5 和 1+6 吗，能否放心地认为它们是正确的？我们感觉 1+5 和 1+6，与前面的 1+1，1+2 都是很类似的简单加法。

等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误（除非等价类中的某些例子属于另一等价类，因为几个等价类是可能相交的）。使用这一方法设计测试用例，首先必须在分析需求规格说明的基础上划分等价类，列出等价类表。

1．划分等价类和列出等价类表

等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的。并合理地假定：测试某等价类的代表值就等于对这一类其他值的测试。

因此，可以把全部输入数据合理地划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件，就可以用少量代表性的测试数据取得较好的测试结果。等价类划分有两种不同的情况：有效等价类和无效等价类。

有效等价类：指对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。

无效等价类：与有效等价类的定义恰巧相反。

设计测试用例时，要同时考虑这两种等价类。因为软件不仅要能接收合理的数据，也要能经受意外的考验。这样的测试才能确保软件具有更高的可靠性。

下面给出6条确定等价类的原则：

① 在输入条件规定了取值范围或值的个数的情况下，可以确立一个有效等价类和两个无效等价类。

② 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可以确立一个有效等价类和一个无效等价类。

③ 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。

④ 在规定了输入数据的一组值（假定n个），并且程序要对每一个输入值分别处理的情况下，可确立n个有效等价类和一个无效等价类。

⑤ 在规定了输入数据必须遵守的规则的情况下，可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

⑥ 在确知已划分的等价类中，各元素在程序处理中的方式不同的情况下，则应再将该等价类进一步地划分为更小的等价类。

在确立了等价类之后，建立等价类表，列出所有划分出的等价类如表5-1所示。

表5-1 等价类表示例

输入条件	有效等价类	无效等价类	输入条件	有效等价类	无效等价类
□	□	□	□	□	□

2．确定测试用例

根据已列出的等价类表，按以下步骤确定测试用例：

① 为每个等价类规定一个惟一的编号。

② 设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。重复这

一步，最后使得所有有效等价类均被测试用例所覆盖。

③ 设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步使所有无效等价类均被覆盖。

在寻找等价区间时，想办法把软件的相似输入、输出、操作分成组。这些组就是等价区间。请看一些例子。

在两数相加用例中，测试 1+13 和 1+99999999 似乎有点不同。这是一种直觉，一个是普通加法，而另一个似乎有些特殊，这个直觉是对的。程序对 1 和最大数值相加的处理和对两个小一些的数值相加的处理有所不同。后者必须处理溢出情况。因为软件操作可能不同，所以这两个用例属于不同的等价区间。

如果具有编程经验，就可能会想到更多可能导致软件操作不同的“特殊”数值。如果不是程序员，也不用担心，你很快就会学到这种技术，无须了解代码细节就可以运用。

如图 5-1 所示是复制的多种方法，给出了选中编辑菜单后显示复制和粘贴命令的计算器程序。每一项功能（即复制和粘贴）有 5 种执行方式。要想复制，可以单击复制菜单命令，键入 C，按 Ctrl+C 或 Ctrl+Shift+C 组合键。任何一种输入途径都会把当前数值复制到剪贴板中，一一执行同样的输出操作，产生同样的结果。

图 5-1 复制的多种方法

如果要测试复制命令，可以把这 5 种输入途径划分减为 3 个，单击菜单命令，键入 C 和按 Ctrl+C 组合键。对软件质量有了信心之后，知道无论以何种方式激活复制功能都工作正常，甚至可以进一步缩减为 1 个区间，例如按 Ctrl+C 组合键。

再看下一个例子。看一下在标准的另存为对话框（如图 5-2 所示）中输入文件名称的情形。

Windows 文件名可以包含除了“、”、“/”、“:”、“ •”、“?”、“<>”和“ ”之外的任意字符。文件名长度是 1～255 个字符。如果为文件名创建测试用例，等价区间有合法字符、非法字符、合法长度的名称、过长名称和过短名称。

例题：根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。“一个程序读入 3 个整数，把这 3 个数值看作一个三角形的 3 条边的长度值。这个程序要打印出信息，说明这个三角形是不等边的、是等腰的、还是等边的”。

我们可以设三角形的 3 条边分别为 A，B，C。如果它们能够构成三角形的 3 条边，必须满足：

A>0，B>0，C>0，且 A+B>C，B+C>A，A+C>B。

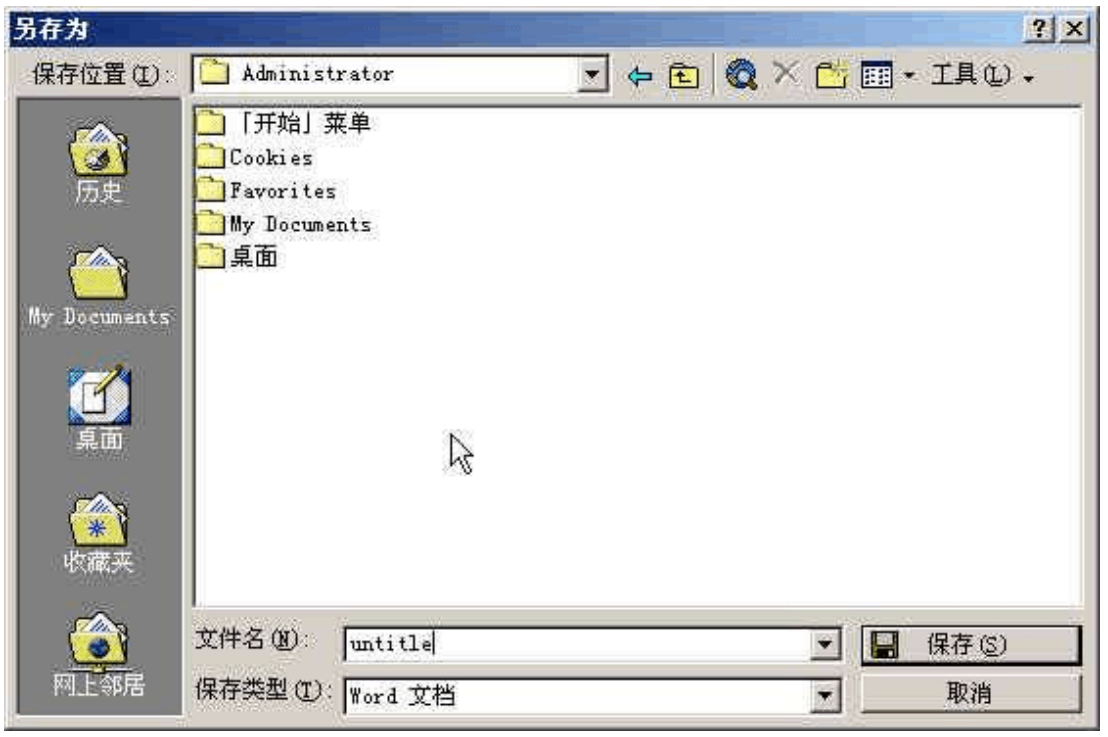

图 5-2　存盘对话框

如果是等腰的，还要判断 A=B，或 B=C，或 A=C。

如果是等边的，则需判断是否 A=B，且 B=C，且 A=C。

列出等价类表，如表 5-2所示。

等 价 类 表

输 入 条 件	有效等价类	无效等价类
是否三角形的 3 条边	(A>0)，（1） (B>0)，（2） (C>0)，（3） (A+B>C)，（4） (B+C>A)，（5） (A+C>B)（6）	(A≤0)，（7） (B≤0)，（8） (C≤0)，（9） (A+B≤C)，（10） (B+C≤A)，（11） (A+C≤B)（12）
是否等腰三角形	(A=B)，（13） (B=C)，（14） (C=A)，（15）	(A≠B) and (B≠C) and (C≠A)，（16）
是否等边三角形	(A=B) and (B=C) and (C=A)，（17）	(A≠B)，（18） (B≠C)，（19） (C≠A)，（20）

设计测试用例：输入顺序是【A，B，C】，如表 5-3所示。

测 试 用 例

序号	【A，B，C】	覆盖等价类	输　　出
1	【3，4，5】	(1)，(2)，(3)，(4)，(5)，(6)	一般三角形
2	【0，1，2】	(7)	不能构成三角形
3	【1，0，2】	(8)	
4	【1，2，0】	(9)	
5	【1，2，3】	(10)	
6	【1，3，2】	(11)	
7	【3，1，2】	(12)	
8	【3，3，4】	(1)，(2)，(3)，(4)，(5)，(6)，(13)	等腰三角形
9	【3，4，4】	(1)，(2)，(3)，(4)，(5)，(6)，(14)	
10	【3，4，3】	(1)，(2)，(3)，(4)，(5)，(6)，(15)	
11	【3，4，5】	(1)，(2)，(3)，(4)，(5)，(6)，(16)	非等腰三角形
12	【3，3，3】	(1)，(2)，(3)，(4)，(5)，(6)，(17)	是等边三角形
13	【3，4，4】	(1)，(2)，(3)，(4)，(5)，(6)，(14)，(18)	非等边三角形
14	【3，4，3】	(1)，(2)，(3)，(4)，(5)，(6)，(15)，(19)	
15	【3，3，4】	(1)，(2)，(3)，(4)，(5)，(6)，(13)，(20)	

请记住，等价分配的目标是把可能的测试用例组合缩减到仍然足以满足软件测试需求为止。因为，选择了不完全测试，就要冒一定的风险，所以必须仔细选择分类。

关于等价分配最后要讲的一点是，这样做有可能不客观。科学有时也是一门艺术。测试同一个复杂程序的两个软件测试员，可能会制定出两组不同的等价区间。只要审查等价区间的人都认为它们足以覆盖测试对象就可以了。

边界值分析法

人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上的，而不是在输入范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。例如，在做三角形计算时，要输入三角形的 3 个边长 A、B 和 C。这 3 个数值应当满足 A>0、B>0、C>0、A+B>C、A+C>B、B+C>A，才能构成三角形。但如果把 6 个不等式中的任何一个大于号“>”错写成大于等于号“≥”，那就不能构成三角形。问题恰恰出现在容易被疏忽的边界附近。这里所说的边界是指相当于输入等价类和输出等价类而言，稍高于其边界值及稍低于其边界值的一些特定情况。

1．边界条件

我们可以想象一下，如果在悬崖峭壁边可以自信地安全行走，平地就不在话下了。如果软件在能力达到极限时能够运行，那么在正常情况下一般也就不会有什么问题。