第5章 黑盒测试技术

5.1 黑盒测试概念

黑盒测试(Black Box Testing)也称功能测试或数据驱动测试。它是已知产品所应具有的功能,通过测试来检测每个功能是否都能正常使用。在测试时,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进行测试,测试者仅依据程序功能的需求规范考虑确定测试用例和推断测试结果的正确性。黑盒测试试图发现以下类型的错误:

(1) 检查程序功能能否按需求规格说明书的规定正常使用,测试各功能是否有遗漏,测试性能等特性是否满足。

(2) 检查人机交互是否正确,检测数据结构或外部数据库访问是否错误,检测程序是否能适当地接收输入数据而产生正确的输出结果,并保持外部信息(如数据库或文件)的完整性。

(3) 检测程序初始化和终止方面的错误。

黑盒测试意味着在软件的接口处进行测试,它着眼于程序外部结构,主要针对软件界面和软件功能进行测试。因此可以说黑盒测试是站在用户的角度,从输入数据与输出结果的对应关系出发进行的测试。黑盒测试的模型如图 5-1 所示。

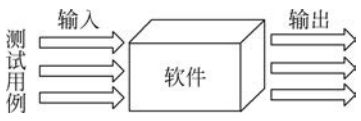


图 5-1 黑盒测试的模型

黑盒测试直观的想法是:既然程序被规定做某些事,就看它是否在任何情况下都做得对。换言之,使用黑盒测试发现程序中的错误,必须在所有可能的输入条件和输出结果下确定测试数据,检查程序是否正确。很显然这是不可能的,因为穷举测试数量太大,不仅要测试所有合法的输入,还要对那些不合法的输入进行测试。因此,黑盒测试的难点在于如何构造有效的测试数据。由于输入空间通常是无限的,穷举测试显然行不通。因此进行黑盒测试时,需要寻找最小最重要的用例集合以精简测试复杂性,提高测试效率。为此黑盒测试也有一套产生测试用例的方法,用以产生有限的测试用例而覆盖足够多的情况。

进行黑盒测试时,根据软件需求规格说明书设计测试用例,在被测试程序上执行测试用例的数据(输入数据/操作步骤),根据输出结果判断程序是否正确,以检测程序是否存在问题。使用黑盒测试技术进行测试的一般过程如图 5-2 所示。

黑盒测试用例设计的主要依据是软件系统需求规格说明书,因此,在进行黑盒测试用例设计之前需要确保说明书是经过评审的,其质量达到了既定的要求。

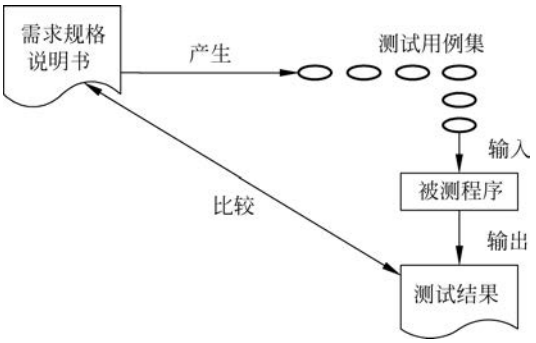


图 5-2 黑盒测试流程

在黑盒测试中，关键的步骤是设计测试用例。常用的黑盒测试用例设计方法有边界值测试、等价类测试、基于判定表的测试、因果图、正交试验法、场景测试法、错误推测法等。下面将详细介绍各测试方法。

5.2 边界值测试

任何一个程序都可以看作是一个函数，程序的输入构成函数的定义域，程序的输出构成函数的值域。人们从长期的测试工作经验中得知，大量的错误是发生在定义域或值域(输出)的边界上，而不是在其内部。对于软件缺陷，有句谚语："缺陷遗漏在角落里，聚集在边界上"。

例如，在做三角形计算时，要输入三角形的三条边长：A、B 和 C。这三个数值应当满足 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。问题常常出现在容易被疏忽的边界附近。类似的例子还有很多，如：计数器常常"少记一次"；循环条件应该是"≤"时错误地写成了"<"；数组下标越界(在 C 语言中数组下标是从零开始，可能错误地认为是从 1 开始，从而使最后一个元素的下标越界)等。

边界值测试背后的基本原理是错误更可能出现在输入变量的极值附近。边界值分析关注的是输入空间的边界，从中标识测试用例。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。

5.2.1 边界条件

边界条件就是一些特殊情况。一般地，在条件 C 下，软件执行一种操作，对任意小的值 σ，在条件 C+σ 或 C−σ 下就会执行另外的操作，则 C 就是一个边界。

在多数情况下，边界条件是基于应用程序的功能设计而需要考虑的因素，可以从软件的规格说明或常识中得到。例如程序要对学生成绩进行处理，要求输入数据的范围是[0，100]，很明显输入条件的边界是 0 和 100。

然而，在测试用例设计过程中，某些边界条件是不需要呈现给用户的，或者说是用户很难注意到的，但同时确实属于检验范畴内的边界条件，称为内部边界条件或次边界条件。

内部边界条件主要有下面几种。

1. 数值的边界值

计算机是基于二进制进行工作的，因此，软件的任何数值运算都有一定的范围限制。例如 1 字节由 8 位组成，1 字节所能表达的数值范围是[0，255]。表 5-1 列出了计算机中常用数值的范围。

表 5-1 二进制数值的边界

术 语	范 围 或 值
bit(位)	0 或 1
byte(字节)	0～255
word(字)	0～65 535(单字)或 0～4 294 967 295(双字)
int(32 位)	－2 147 483 648～2 147 483 647
K(千)	1024
M(兆)	1 048 576
G(千兆)	1 073 741 824

2. 字符的边界值

在计算机软件中，字符也是很重要的表示元素。其中，ASCII 和 Unicode 是常见的编码方式。表 5-2 中列出了一些常用字符对应的 ASCII 码值。如果要测试文本输入或文本转换的软件，在定义数据区间包含哪些值时，就可以参考以下 ASCII 码表，找出隐含的边界条件。

表 5-2 部分 ASCII 码值表

字 符	ASCII 码值	字 符	ASCII 码值
Null(空)	0	A	65
Space(空格)	32	a	97
/(斜杠)	47	Z	90
0(零)	48	z	122
:(冒号)	58	‘(单引号)	96
@	64	{(大括号)	123

3. 其他边界条件

有一些边界条件容易被人忽略，如在文本框中不是没有输入正确的信息，而是根本就没有输入任何内容，然后就单击“确认”按钮。这种情况常常被遗忘或忽视了，但在实际使用中却时常发生。因此在测试时还需要考虑程序对默认值、空白、空值、零值、无输入等情况。

在进行边界值测试时，如何确定边界条件的取值呢？一般情况下，确定边界值应遵循以下几条原则。

(1) 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。

（2）如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少 1、比最大个数多 1 的数作为测试数据。

（3）如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试数据。

（4）如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试数据。

（5）分析规格说明，找出其他可能的边界条件。

5.2.2　边界值分析

为便于理解，以下讨论涉及两个输入变量 x_1 和 x_2 的函数 F。假设 x_1 和 x_2 分别在下列的范围内取值：a≤x_1≤b；c≤x_2≤d。函数 F 的输入空间如图 5-3 所示。阴影矩形中的任何一点都是函数 F 的有效输入。

边界值分析的基本思想是使用输入变量的最小值、略高于最小值、正常值、略低于最大值和最大值设计测试用例。通常用 min、min＋、nom、max－和 max 来表示。

当一个函数或程序有两个及两个以上的输入变量时，就需要考虑如何组合各变量的取值。可根据可靠性理论中的单缺陷假设和多缺陷假设来考虑。单缺陷假设是指“失效极少是由两个或两个以上的缺陷同时发生引起的”。因此依据单缺陷假设来设计测试用例，只需让一个变量取边界值，其余变量取正常值。多缺陷假设是指“失效是由两个或两个以上缺陷同时作用引起的”。因此依据多缺陷假设来设计测试用例，要求在选取测试用例时同时让多个变量取边界值。

在边界值分析中，用到了单缺陷假设，即选取测试用例时仅仅使得一个变量取极值，其他变量均取正常值。对于有两个输入变量的程序 P，其边界值分析的测试用例如下：

$\{<x_{1nom},x_{2min}>,<x_{1nom},x_{2min+}>,<x_{1nom},x_{2nom}>,<x_{1nom},x_{2max-}>,<x_{1nom},x_{2max}>,<x_{1min},x_{2nom}>,<x_{1min+},x_{2nom}>,<x_{1max-},x_{2nom}>,<x_{1max},x_{2nom}>\}$

对于有两个输入变量的程序 P，其边界值分析的测试用例在图中的位置如图 5-4 所示。

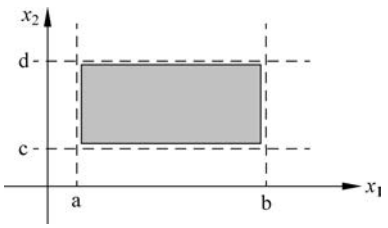


图 5-3　两个变量函数的输入域

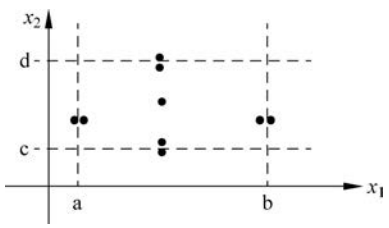


图 5-4　两个变量函数的边界值分析

例如，有一个二元函数 $f(x,y)$，要求输入变量 x,y 分别满足：$x\in[1,12]$，$y\in[1,31]$。采用边界值分析法设计测试用例，可以选择下面一组测试数据：{ <1,15>,<2,15>,<11,15>,<12,15>,<6,15>,<6,1>,<6,2>,<6,30>,<6,31> }。

对于一个含有 n 个输入变量的程序，使除一个以外的所有变量取正常值，使剩余的那个变量依次取最小值、略高于最小值、正常值、略低于最大值和最大值，对每个变量都重复进行。因此，对于有 n 个输入变量的程序，边界值分析会产生 $4n+1$ 个测试用例。

例如，有一个三元函数 $f(x, y, z)$，其中 $x \in [0,100]$，$y \in [1,12]$，$z \in [1,31]$，对该函数采用边界值分析法设计的测试用例将会得到 13 个测试用例，根据边界分析的原理，可得到下列测试数据：{ <50,6,1>,<50,6,2>,<50,6,30>,<50,6,31>,<50,1,15>,<50,2,15>,<50,11,15>,<50,12,15>,<0,6,15>,<1,6,15>,<99,6,15>,<100,6,15>,<50,6,15> }。

5.2.3 健壮性边界测试

健壮性是指在异常情况下，软件还能正常运行的能力。健壮性可衡量软件对于规范要求以外的输入情况的处理能力。所谓健壮的系统，是指对于规范要求以外的输入能够判断出这个输入不符合规范要求，并能有合理的处理方式。软件设计的健壮与否直接反映了分析设计和编码人员的水平。

健壮性边界测试是边界值分析的一种简单扩展。在使用该方法设计测试用例时，既要考虑有效输入，也要考虑无效的输入。除了按照边界值分析方法选取的五个取值（min、min＋、nom、max－、max），还要选取略小于最小值（min－）和略大于最大值（max＋）的取值，以观察输入变量超过边界时程序会有什么表现。对于有两个变量的程序 P，其健壮性测试的测试用例如图 5-5 所示。

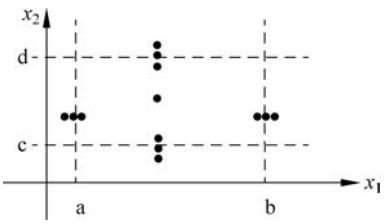


图 5-5 两个变量函数的健壮性测试用例

对于一个输入变量为 n 的程序，进行健壮性边界测试时，使除一个以外的所有变量取正常值，使剩余的那个变量依次取略小于最小值、最小值、略高于最小值、正常值、略低于最大值、最大值和略高于最大值，并对每个变量重复进行。因此其健壮性测试会产生 $6n+1$ 个测试用例。

例如，有一个二元函数 $f(x,y)$，要求输入变量 x，y 分别满足：$x \in [0,100]$，$y \in [1000,3000]$，对其进行健壮性测试，则需要设计 13 个测试用例。根据健壮性测试的原理，可以得到下面一组测试数据：{<－1,1500>,<0,1500>,<1,1500>,<50,1500>,<99,1500>,<100,1500>,<101,1500>,<50,999>,<50,1000>,<50,1001>,<50,2999>,<50,3000>,<50,3001>}。

健壮性测试最关心的是预期的输出，而不是输入。健壮性测试的最大价值在于观察处理异常情况，它是检测软件系统容错性的重要手段。

5.2.4 最坏情况测试

最坏情况测试拒绝单缺陷假设，它关心的是当多个变量取极值时出现的情况。最坏情况测试中，对每一个输入变量首先获得包含最小值、略高于最小值、正常值、略低于最大值、最大值的 5 个元素集合的测试，然后对这些集合进行笛卡儿积计算，以生成测试用例。

对于有两个变量的程序 P，其最坏情况测试的测试用例如图 5-6 所示。

显然，最坏情况测试更加彻底，因为边界值分析测试是最坏情况测试用例的真子集。进行最坏情况测试意味着更多的测试工作量：n 个变量的函数，其最坏情况测试将会产生 5^n 个测试用例，而边界值分析只产生 $4n+1$ 个测试用例。

健壮最坏情况测试是最坏情况测试的扩展，这种测试使用健壮性测试的 7 个元素集合的笛卡儿积，将会产生 7^n 个测试用例。图 5-7 给出了两个变量函数的最坏情况测试用例。

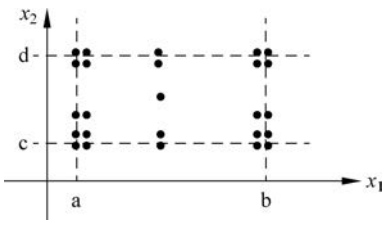


图 5-6 两个变量函数的最坏情况测试用例

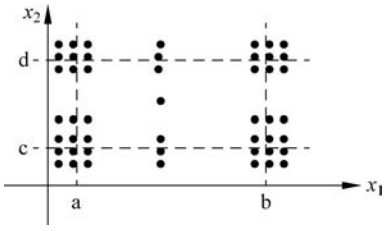


图 5-7 两个变量函数的健壮最坏情况测试用例

例 5-1 NextDate 程序。

程序有三个输入变量 month、day、year(month、day 和 year 均为整数值，并且满足：1≤month≤12，1≤day≤31，1900≤year≤2050)，分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上下一天的日期。例如，输入为 2000 年 12 月 31 日，则该程序的输出为 2001 年 1 月 1 日。请用健壮性测试法设计测试用例。

下面用健壮性测试法设计测试用例，按照下列步骤进行：

(1) 分析各变量的取值。

健壮性测试时，各变量分别取：略小于最小值、最小值、略高于最小值、正常值、略低于最大值、最大值和略大于最大值。

month：－1，1，2，6，11，12，13。

day：－1，1，2，15，30，31，32。

year：1899，1900，1901，1975，2049，2050，2051。

(2) 测试用例数。

有 n 个变量的程序，其边界值分析会产生 $6n+1$ 个测试用例。这里有 3 个变量，因此会产生 19 个测试用例。

(3) 设计测试用例，如表 5-3 所示。

表 5-3 NextDate 函数测试用例

测试用例	输入数据			预 期 输 出
	mouth	day	year	
1	6	15	1899	year 超出[1900，2050]
2	6	15	1900	1900.6.16
3	6	15	1901	1901.6.16
4	6	15	1975	1975.6.16
5	6	15	2049	2049.6.16
6	6	15	2050	2050.6.16
7	6	15	2051	year 超出[1900，2050]
8	6	－1	1975	day 超出[1…31]
9	6	1	1975	1975.6.2
10	6	2	1975	1975.6.3

续表

测试用例	输入数据			预 期 输 出
	mouth	day	year	
11	6	30	1975	1975.7.1
12	6	31	1975	输入日期超界
13	6	32	1975	day 超出[1…31]
14	−1	15	1975	Mouth 超出[1…12]
15	1	15	1975	1975.1.16
16	2	15	1975	1975.2.16
17	11	15	1975	1975.11.16
18	12	15	1975	1975.12.16
19	13	15	1975	Mouth 超出[1…12]

NextDate 函数的复杂性来源于两个方面：一是输入域的复杂性(即输入变量之间逻辑关系的复杂性)，二是确定闰年的规则。但是在进行健壮性测试时，没有考虑输入变量之间的逻辑关系，也没有考虑和闰年相关的问题，因此在设计测试用例时存在遗漏问题，例如和判断闰年相关的日期：2008.2.29、1999.2.28 等。

5.3 等价类测试

软件测试有一个致命的缺陷，即测试的不完全和不彻底性。任何程序只能进行少量(相对于穷举的巨大数量而言)的和有限的测试。在测试时，既要考虑到测试的效果，又要考虑到软件测试的经济性，为此我们引入等价类的思想。使用等价类划分的目的是在有限的测试资源的情况下，用少量有代表性的数据得到比较好的测试效果。

等价类测试是把所有可能的输入数据，即程序的输入域划分成若干部分(子集)，然后从每一个子集中选取少数具有代表性的数据作为测试用例。该方法是一种重要的、常用的黑盒测试用例设计方法。

5.3.1 等价类

1. 等价类的划分

等价类的重要问题是它们构成集合的划分。划分是指互不相交的一组子集，这些子集的并是整个集合。划分可定义为：给定集合 B，以及 B 的一组子集 A_1、A_2、…、A_n，这些子集是 B 的一个划分，当且仅当 $A_1 \cup A_2 \cup \cdots \cup A_n = B$，且 $i \neq j$ 时 $A_i \cap A_j = \Phi$。

划分对于测试有非常重要的意义：①各个子集的并是整个集合，这提供了一种形式的完备性；②各个子集的交是空，这种互不相交保证了一种形式的无冗余性。因此采用划分可保证某种程度的完备性，并减少冗余。

等价类划分是将输入定义域进行一个划分，并且划分的各个子集是由等价关系决定的。这里的等价关系是指：在子集合中，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个类中其他值的测试。也就是说，如果等

价类中某个输入条件不能导致问题发生，那么用该等价类中其他输入条件进行测试也不可能发现错误。

等价类划分有两种不同的情况：有效等价类和无效等价类。

有效等价类是指对于程序的规格说明是合理的、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。无效等价类是指对程序的规格说明是不合理的或无意义的输入数据所构成的集合。对于具体的问题，无效等价类至少应有一个，也可能有多个。

在设计测试用例时，要同时考虑这两种等价类。因为用户在使用软件时，有意或无意输入一些非法的数据是常有的事情。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。

2．划分等价类的方法

等价类测试的思想就是把全部输入数据合理划分为若干等价类，在每一个等价类中取一个具有代表性的数据作为测试的输入条件，这样可以用少量的测试数据取得较好的测试效果。

在等价类测试中，划分等价类是非常关键的。如果等价类划分合理，可以大大减少测试用例，并能保证达到要求的测试覆盖率。那如何划分等价类呢？一般来讲，等价类划分首先要分析程序所有可能的输入情况，然后按照下列规则对其进行划分。

（1）按照区间划分。在输入条件规定了取值范围或值的个数的情况下，则可以确立 1 个有效等价类和 2 个无效等价类。例如，程序的输入是学生成绩，其范围是 0～100，则输入条件的等价类如图 5-8 所示。

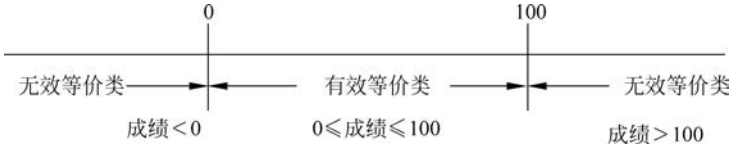


图 5-8　学生成绩的等价类

其有效等价类为 0≤成绩≤100；无效等价类为成绩＜0，成绩＞100。

（2）按照数值划分。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 1 个无效等价类。

例如，程序输入 x 取值于一个固定的枚举类型{1,2,4,12}，并且程序中对这 4 个数值分别进行了处理，则有效等价类为 $x=1$，$x=2$，$x=4$，$x=12$，无效等价类为 1,2,4,12 以外的值构成的集合。

又如，在教师上岗方案中规定对教授、副教授、讲师和助教分别处理。那么可以确定 4 个有效等价类：教授、副教授、讲师和助教；1 个无效等价类，它是所有不符合以上职称的人员构成的集合。

（3）按照数值集合划分。在输入条件规定了输入值的集合或者规定了“必须如何”的情况下，可确立 1 个有效等价类和 1 个无效等价类。例如，某程序中有标识符，其输入条件规

定“标识符应以字母开头……”,则可以这样划分等价类:“以字母开头者”作为有效等价类,“以非字母开头”作为无效等价类。

(4) 在输入条件是一个布尔量的情况下,可确定 1 个有效等价类和 1 个无效等价类。例如验证码,在登录各种网站时经常使用。验证码是一种布尔型取值,True 或者 False。在这里可划分出 1 个有效等价类和 1 个无效等价类。

(5) 进一步细分等价类。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。例如,程序用于判断几何图形的形状,则可以首先根据边数划分出三角形、四边形、五边形、六边形等。然后对于每一种类型,做进一步的划分,例如三角形可以进一步分为等边三角形、等腰三角形、一般三角形。

(6) 等价类划分还应特别注意默认值、空值、NULL、0 等的情形。

3. 等价类的特点

按划分等价类的规则划分出的等价类具有下列特点:

(1) 完备性。划分出的各个等价类(子集)的并是输入/输出的全集,即程序的定义域/值域。

(2) 无冗余性。各个等价类是互不相交的一组子集。

(3) 等价性。划分的各个子集是由等价关系决定的,即各个输入数据对于揭露程序中的错误都是等效的。因此我们可以从等价类中选择一个具有代表性的数据进行测试就可以达到测试目的。

5.3.2 等价类测试类型

等价类划分既实现了完备性测试,又避免了冗余。在实际使用等价类方法测试时,需要考虑等价类测试的程度,不同程度的测试将得到不同的测试效果。基于单缺陷假设还是基于多缺陷假设,产生弱等价类与强等价类测试之分;是否考虑无效等价类(即是否进行无效数据的处理),产生健壮与一般等价类测试之分。等价类测试分为 4 种形式:弱一般等价类、强一般等价类、弱健壮等价类和强健壮等价类。

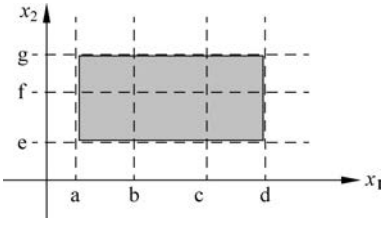


图 5-9 两个变量的边界及边界内区间划分

为便于讨论,下面以一个具有两个变量 x_1 和 x_2 的函数 F 为例。输入变量 x_1 和 x_2 的边界以及边界内的区间为:a$\leqslant x_1 \leqslant$d,区间为[a,b),[b,c),[c,d];e$\leqslant x_2 \leqslant$g,区间为[e,f),[f,g]。

此函数的区间划分如图 5-9 所示。

1. 弱一般等价类测试

弱一般等价类测试遵循单缺陷假设,要求选取的测试用例覆盖所有的有效等价类。两个变量的弱一般等价类测试用例如图 5-10 所示。

2. 强一般等价类测试

强一般等价类测试基于多缺陷假设,要求将每个变量的有效等价类做笛卡儿积,设计测

试用例覆盖笛卡儿积的每个元素。两个变量的强一般等价类测试用例如图 5-11 所示。

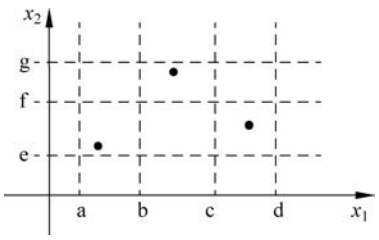


图 5-10 弱一般等价类测试用例

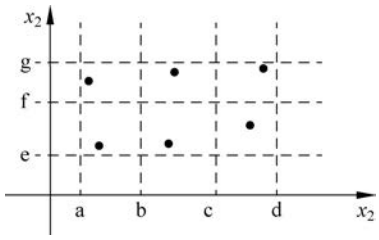


图 5-11 强一般等价类测试用例

3. 弱健壮等价类测试

这里的“弱”是指等价类测试基于单缺陷假设，而“健壮”是考虑了无效值。采用弱健壮等价类测试时，对有效输入，使用每个有效等价类的一个值；对无效输入，测试用例将拥有一个无效值，并保持其余的值都是有效的。

进行弱健壮等价类测试也就是将弱一般等价类中的 5 个要素增添为 7 个要素，补充输入域边界以外的值(略小于最小值 min－1，略大于最大值 max＋1)，涵盖了有效测试和无效测试。两个变量的弱健壮等价类测试用例如图 5-12 所示。

4. 强健壮等价类测试

强健壮等价类测试是基于多缺陷假设，并考虑无效的输入。设计测试用例时需要从所有等价类的笛卡儿积的每一个元素中获得测试用例。两个变量的强健壮等价类测试用例如图 5-13 所示。

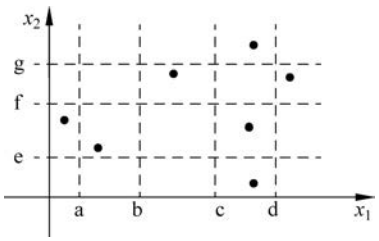


图 5-12 弱健壮等价类测试用例

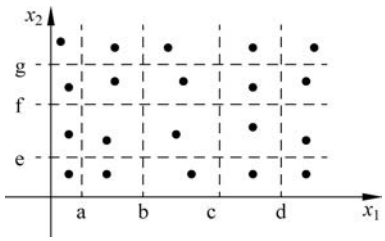


图 5-13 强健壮等价类测试用例

健壮等价类测试存在两个问题：

(1) 需要花费精力定义无效测试用例的期望输出；

(2) 对强类型的语言没有必要考虑无效的输入。

5.3.3 用等价类设计测试用例

1. 划分等价类

首先根据输入条件或输出条件划分等价类。

2. 建立等价类表

根据划分的等价类建立等价类表。

3. 选择测试用例

从等价类中选取具有代表性的数据，设计测试用例。从等价类中选择测试用例时，一般遵循下列原则：

(1) 为每一个等价类规定一个唯一的编号。

(2) 设计一个测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。重复这一步，直到所有的有效等价类都被覆盖为止。

(3) 设计一个测试用例，使其仅覆盖一个尚未被覆盖的无效等价类。重复这一步，直到所有的无效等价类都被覆盖为止。

这里强调每次只覆盖一个无效等价类。这是因为一个测试用例中如果含有多个缺陷，有可能在测试中只发现其中的一个，另一些被忽视。等价类划分法能够全面、系统地考虑黑盒测试的测试用例设计问题，但是没有考虑各变量之间的逻辑关系。

5.3.4 等价类测试指导方针

等价类测试的一些观察和等价类测试的指导方针如下。

(1) 等价类测试的弱形式不如对应的强形式的测试全面。

(2) 如果实现语言是强类型，则没有必要使用健壮形式的测试。

(3) 如果错误条件非常重要，则进行健壮形式的测试是合适的。

(4) 如果输入数据以离散值区间和集合定义，则等价类测试是合适的。当然也适用于如果变量值越界就会出现故障的系统。

(5) 通过结合边界值测试，等价类测试可得到加强。

(6) 如果程序函数很复杂，则等价类测试是被指示的。在这种情况下，函数的复杂性可以帮助标识有用的等价类。

(7) 强等价类测试假设变量是独立的，相应的测试用例相乘会引起冗余问题。如果存在依赖关系，则常常会生成错误测试用例。

(8) 在发现合适的等价关系之前，可能需要进行多次尝试。

(9) 强和弱形式的等价类测试之间的差别，有助于区分累进测试和回归测试。

例 5-2 排序问题。

某程序的功能是输入一组整型数据(数据个数不超过 100 个)，使用冒泡排序法进行排序，数据按从小到大的顺序排列。

下面用等价类方法设计测试用例。

(1) 划分等价类。根据程序的功能要求可以从下列几个方面划分等价类：数据类型；数据个数；数据是否有序；数据是否相同。

排序问题的等价类划分如表 5-4 所示。

(2) 设计测试用例。根据表 5-4 中的等价类设计测试用例，如表 5-5 所示。

表 5-4 排序问题的等价类划分

有效等价类	编号	无效等价类	编号
整数	1	小数	2
		非数值类型的字符	3
1 个整数	4	0 个整数	5
100 以内的多个整数(包括 100 个)	6	多余 100 个整数	7
多个(100 个以内)无序的整数	8		
多个(100 个以内)已按从小到大排好序的整数	9		
多个(100 个以内)已按从大到小排好序的整数	10		
多个(100 个以内)相同的数据	11		

表 5-5 排序问题的测试用例

编 号	输 入	预 期 输 出	覆盖等价类
1	5	5	1,4
2	0.5	提示：请输入整数	2
3	a b	提示：请输入整数	3
4	空	提示：请输入数据	5
5	1,4,2,8,11,6	1,2,4,6,8,11	1,6,8
6	1,2,… ,110(110 个数据)	提示：数据太多	1,7
7	1,2,3,4,5,6	1,2,3,4,5,6	1,6,9
8	6,5,4,3,2,1	1,2,3,4,5,6	1,6,10
9	5,5,5,5,5	5,5,5,5,5	1,6,11

5.4 基于判定表的测试

在一些数据处理问题中,某些操作是否实施依赖于多个逻辑条件的取值。在这些逻辑条件取值的组合所构成的多种情况下,分别执行不同的操作。处理这类问题的一个非常有力的分析和表达工具是判定表,或称决策表(Decision Table)。判定表能够将复杂的问题按照各种可能的情况全部列举出来,简明并避免遗漏。因此,利用判定表能够设计出完整的测试用例集合。在所有功能性测试方法中,基于判定表的测试方法是最严格的,因为判定表在逻辑上是严密的。

5.4.1 判定表的组成

判定表通常由 4 个部分组成,如图 5-14 所示。

(1) 条件桩(Condition Stub)。列出了问题的所有条件。通常认为列出的条件的次序无关紧要。

(2) 动作桩(Action Stub)。列出了问题规定可能采取的操作。这些操作的排列顺序没有约束。

(3) 条件项(Condition Entry)。列出针对其左

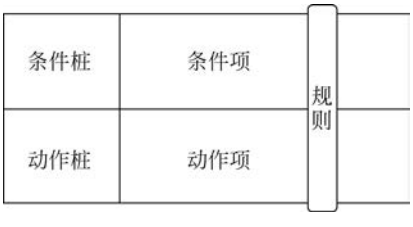


图 5-14 判定表结构

侧条件的取值。

(4) 动作项(Action Entry)。列出在条件项的各种取值情况下应该采取的动作。

动作项和条件项紧密相关,它指出了在条件项的各组取值情况下应采取的动作。任何一个条件组合的特定取值及其相应要执行的操作称为规则。在判定表中贯穿条件项和动作项的一列就是一条规则。规则指示了在各条件项指示的条件下要采取动作项中的行为。显然,判定表中列出多少个条件取值,也就有多少条规则,即条件项和动作项有多少列。

判定表可分为有限条目判定表和扩展条目判定表。有限条目判定表的特点是:所有条件都是二值条件(真/假),有 n 个条件的判定表将有 $2n$ 条规则。扩展条目判定表的特点是:条件可以有多个值,有 n 个条件的判定表,其规则条数为 n 个条件所有值的笛卡儿积。

下面通过一个例子来说明判定表各部分的含义。

在表 5-6 给出的判定表中,规则 1 表示:如果条件 1、条件 2、条件 3 分别为真,则采取动作 1 和动作 2。规则 2 表示:如果条件 1 和条件 2 为真,条件 3 为假,则采取动作 3。我们注意到在表 5-6 的规则 5 中,条件 3 用"—"表示,意思是条件 3 为不关心条目。不关心条目有两种主要解释:条件无关或条件不适用。规则 5 表示:条件 1 为假、条件 2 为真时,则采取动作 2,而不管条件 3 为真还是为假,或者条件 3 不适用。

表 5-6 判定表实例

桩	规则 1	规则 2	规则 3	规则 4	规则 5	规则 6	规则 7
条件 1	T	T	T	T	F	F	F
条件 2	T	T	F	F	T	F	F
条件 3	T	F	T	F	—	T	F
动作 1	✓		✓	✓			
动作 2	✓				✓	✓	
动作 3		✓		✓		✓	
动作 4							✓

在实际使用判定表时,通常要将其化简。化简工作是以合并相似规则为目标。若表中有两条或多条规则具有相同的动作,并且其条件项之间存在着极为相似的关系,便可设法将其合并。例如,在图 5-15(a)中左端的两规则其动作项一致,条件项中第 1、2 条件项取值一致,只是第 3 条件项取值不同。这一情况表明,在第 1、2 条件项分别取真值和假值时,无论第 3 条件取任何值,都要执行同一操作。即要执行的动作与第 3 条件项的取值无关。因此,

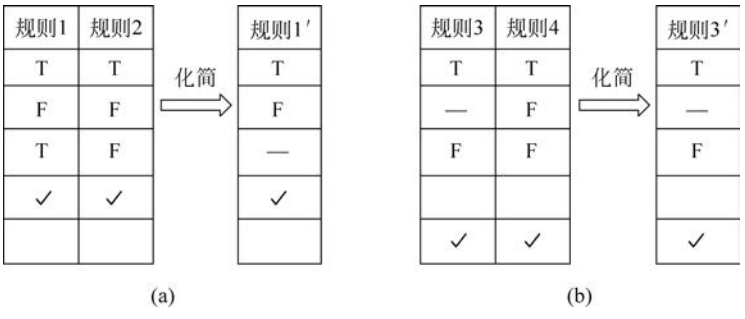


图 5-15 规则合并

可将这两个规则合并。合并后的第 3 条件项用特定的符号“—”表示与取值无关。

与此类似,无关条件项“—”在逻辑上又可包含其他的条件项取值,具有相同动作的规则还可进一步合并,如图 5-15(b)所示。

5.4.2 基于判定表的测试

为了使用判定表标识测试用例,在这里把条件解释为程序的输入,把动作解释为输出。在测试时,有时条件最终引用输入的等价类,动作引用被测程序的主要功能处理,这时规则就解释为测试用例。由于判定表的特点,可以保证能够取到输入条件的所有可能的条件组合值,因此可以做到测试用例的完整集合。

使用判定表进行测试时,首先需要根据软件规格说明建立判定表。判定表设计的步骤如下。

(1) 确定规则的个数。根据被测试软件的特点来选择判定表的类型。如果采用有限条目的判定表,假如有 n 个条件,则会产生 2^n 条规则。如果采用扩展条目判定表,则规则数等于 n 个条件所有值的笛卡儿积。

(2) 列出所有的条件桩和动作桩。在测试中,条件桩一般对应程序输入的各个条件项,而动作桩一般对应程序的输出结果或要采取的操作。

(3) 填入条件项。条件项就是每条规则中各个条件的取值。为了保证条件项取值的完备性和正确性,可以利用集合的笛卡儿积来计算。首先找出各条件项取值的集合,然后将各集合做笛卡儿积,最后将得到的集合的每一个元素填入规则的条件项中。

(4) 填入动作项,得到初始判定表。在填入动作项时,必须根据程序的功能说明来填写。首先根据每条规则中各条件项的取值,来获得程序的输出结果或应该采取的行动,然后在对应的动作项中做标记。

(5) 简化判定表、合并相似规则(相同动作)。若表中有两条以上规则具有相同的动作,并且在条件项之间存在极为相似的关系,便可以合并。合并后的条件项用符号“—”表示,说明执行的动作与该条件的取值无关,称为无关条件。

5.4.3 基于判定表测试的指导方针

基于判定表的测试能把复杂的问题按各种可能的情况一一列举出来,简明而易于理解,也可避免遗漏。但是,判定表不能表达重复执行的动作,例如循环结构。

与其他测试技术一样,基于判定表的测试对于某些应用程序很有效,对于另一些应用程序却不适用。B. Beizer 指出了适合使用判定表设计测试用例的条件。

(1) 规格说明以判定表形式给出,或很容易转换成判定表。

(2) 条件的排列顺序不会也不影响执行哪些操作。

(3) 规则的排列顺序不会也不影响执行哪些操作。

(4) 每当某一规则的条件已经满足,并确定要执行的操作后,不必检验别的规则。

(5) 如果某一规则得到满足要执行多个操作,这些操作的执行顺序无关紧要。

B. Beizer 提出这 5 个必要条件的目的是使操作的执行完全依赖于条件的组合。其实对于某些不满足这几条的判定表,同样可以借助这种方法设计测试用例,只不过尚需增加其

他的测试用例罢了。

判定表对于有 if else 或者 switch case 的程序，设计测试用例时非常有帮助。它在多数情况下是一种理清思路的工具，比流程图更为直观，可以写出符合需求说明的测试用例。

例 5-3 考生录取问题。

某程序规定："对总成绩大于 450 分，且各科成绩均高于 85 分或者是优秀毕业生，应优先录取，其余情况做其他处理"。请用判定表测试方法设计测试用例。

根据被测试对象的特点，采用有限条目判定表设计测试用例，具体步骤如下。

(1) 列出所有的条件桩和动作桩。根据问题描述的输入条件和输出结果，列出所有的条件桩和动作桩。

输入条件(条件桩)：

① 总成绩大于 450 分吗?

② 各科成绩均高于 85 分吗?

③ 是优秀毕业生吗?

输出结果(动作桩)：

① 优先录取；

② 做其他处理。

(2) 确定规则的个数。本例中输入有三个条件，每个条件的取值为"是"或"否"，因此有 2×2×2=8 种规则。

(3) 填入条件项。在填写条件项时，可以将各个条件取值的集合进行笛卡儿积，得到每一列条件项的取值。本例就是计算{Y,N}×{Y,N}×{Y,N}={<Y,Y,Y>,<Y,Y,N>,<Y,N,Y>,<Y,N,N>,<N,Y,Y>,<N,Y,N>,<N,N,Y>,<N,N,N>}，然后将所得集合中的每一个元素的值填入每一列各条件项中，如表 5-7 所示。

表 5-7 判定表

		1	2	3	4	5	6	7	8
条件	总成绩大于 450 分吗	Y	Y	Y	Y	N	N	N	N
	各科成绩均高于 85 分吗	Y	Y	N	N	Y	Y	N	N
	是优秀毕业生吗	Y	N	Y	N	Y	N	Y	N
动作	优先录取	√	√	√					
	做其他处理				√	√	√	√	√

(4) 填入动作桩和动作顶。根据每一列中各条件的取值得到所要采取的行动，填入动作桩和动作项，便得到初始判定表，如表 5-7 所示。

(5) 化简。从表 5-7 中，可以很直观地看出规则 1 和规则 2 的动作项相同，第 1 条件项和第 2 条件项的取值相同，只有第 3 条件项的取值不同，满足合并的原则。合并时，第 3 条件项成为不相关条目，用"—"表示。同理，规则 5 和规则 6 可以合并，规则 7 和规则 8 可以合并。通过合并相似规则后得到简化的判定表，如表 5-8 所示。

从表 5-8 可以看出规则 4 和规则 5 还可以进一步合并，合并后的判定表如表 5-9 所示。

表 5-8 简化后的判定表

		1	2	3	4	5
条件	总成绩大于 450 分吗	Y	Y	Y	N	N
	各科成绩均高于 85 分吗	Y	N	N	Y	N
	是优秀毕业生吗	—	Y	N	—	—
动作	优先录取	✓	✓			
	做其他处理			✓	✓	✓

表 5-9 进一步简化后的判定表

		1	2	3	4
条件	总成绩大于 450 分吗	Y	Y	Y	N
	各科成绩均高于 85 分吗	Y	N	N	—
	是优秀毕业生吗	—	Y	N	—
动作	优先录取	✓	✓		
	做其他处理			✓	✓

例 5-4 NextDate 程序。

NextDate 程序的描述见 5.2 节例 5-1。下面用判定表测试法进行测试用例设计。

问题分析：年、月、日三个变量之间在输入定义域中存在一定的逻辑依赖关系，由于等价类划分和边界值分析测试都假设了变量是独立的，如果采用边界值测试或等价类测试方法设计测试用例，那么这些依赖关系在机械地选取输入值时可能会丢失。而采用判定表法则可以通过使用“不可能”的概念表示条件的不可能组合，来强调这种依赖关系。

根据题目的要求，下面用基于判定表的方法进行测试。

(1) 分析各种输入情况，列出为输入变量 month、day、year 划分的有效等价类。

month 变量的有效等价类：

M1={month=4,6,9,11};

M2={month=1,3,5,7,8,10};

M3={month=12};

M4={month=2}。

day 变量的有效等价类：

D1={1≤day≤27};

D2={day=28};

D3={day=29};

D4={day=30};

D5={day=31}。

year 变量的有效等价类：

Y1={year 是闰年};

Y2={year 是平年}。

(2) 分析程序规格说明，结合以上等价类划分的情况给出问题规定的可能采取的操作(即列出所有的动作桩)。考虑各种有效的输入情况，程序中可能采取的操作有以下 6 种。

a1：day＋1；

a2：day＝1；

a3：month＋1；

a4：month＝1；

a5：year＋1；

a6：不可能。

(3) 根据步骤(1)和步骤(2)，画出判定表，然后对判定表进行化简。简化后的判定表如表 5-10 所示。

表 5-10　NextDate 问题的判定表

		1	2	3	4	5	6	7
条件	月份属于	M1	M1	M1	M2	M2	M3	M3
	日期属于	D1，D2，D3	D4	D5	D1，D2，D3，D4	D5	D1，D2，D3，D4	D5
	年份属于	—	—	—	—	—	—	—
动作	a1：day＋1	√			√			
	a2：day＝1		√			√		√
	a3：month＋1		√			√		
	a4：month＝1							√
	a5：year＋1							√
	a6：不可能			√				

		8	9	10	11	12	13
条件	月份属于	M4	M4	M4	M4	M4	M4
	日期属于	D1	D2	D2	D3*	D3	D4，D5
	年分属于	—	Y1	Y2	Y1	Y2	—
动作	a1：day＋1	√	√				
	a2：day＝1			√	√		
	a3：month＋1			√	√		
	a4：month＝1		√				
	a5：year＋1		√				
	a6：不可能					√	√

(4) 设计测试用例。为判定表中的每一列设计一个测试用例，如表 5-11 所示。

表 5-11　隔一日问题测试用例

测试用例编号	输入数据			预期结果	覆盖的规则
	月份	日期	年		
1	6	15	2005	2005 年 6 月 16 日	1
2	9	30	2005	2005 年 10 月 1 日	2
3	4	31	2011	提示用户输入错误	3
4	1	1	2005	2005 年 1 月 2 日	4
5	3	31	2000	2000 年 4 月 1 日	5
6	12	5	1999	1999 年 12 月 6 日	6

续表

测试用例编号	输入数据			预期结果	覆盖的规则
	月份	日期	年		
7	12	31	1999	2000年1月1日	7
8	2	26	1999	1999年2月27日	8
9	2	28	2000	2000年2月29日	9
10	2	28	2001	2001年3月1日	10
11	2	29	2000	2000年3月1日	11
12	2	29	2005	提示用户输入错误	12
13	2	30	2000	提示用户输入错误	13

5.5 因果图

前面介绍的等价类划分法和边界值分析方法都是着重考虑输入条件，但没有考虑输入条件的各种组合和输入条件之间的相互制约关系。这样虽然各种输入条件可能出错的情况已经测试到了，但多个输入条件组合起来可能出错的情况却被忽视了。如果考虑输入条件之间的相互组合，可能会产生一些新的情况。但要检查输入条件的组合不是一件容易的事情，即使把所有输入条件划分成等价类，它们之间的组合情况也相当多。因此必须考虑采用一种适合于描述对于多种条件的组合，相应产生多个动作的形式来考虑设计测试用例，这就需要利用因果图(逻辑模型)。因果图方法最终生成的就是判定表，它适合于检查程序输入条件的各种组合情况。

5.5.1 因果图的概念

20世纪70年代，IBM进行了一项工作，把自然语言书写的需求转换成一个形式说明，形式说明可以用来产生功能测试的测试实例。这个转换过程检查需求的语义，用输入和输出之间或输入和转换之间的逻辑关系来重新表述它们。输入称为原因，输出和转换称为结果。通过分析得到一张反映这些关系的图，称为因果图(Cause-and-Effect Graph)。

因果图中使用了简单的逻辑符号，以直线连接左右节点。左节点表示输入状态(或称原因)，右节点表示输出状态(或称结果)。通常用 c_i 表示原因，一般置于图的左部；e_i 表示结果，通常在图的右部。c_i 和 e_i 均可取值“0”或“1”，其中“0”表示某状态不出现，“1”表示某状态出现。

因果图中包含以下4种关系。

(1) 恒等。若 c_1 是1，则 e_1 也是1；若 c_1 是0，则 e_1 为0。

(2) 非。若 c_1 是1，则 e_1 是0；若 c_1 是0，则 e_1 是1。

(3) 或。若 c_1 或 c_2 或 c_3 是1，则 e_1 是1；若 c_1、c_2 和 c_3 都是0，则 e_1 为0。“或”可有任意多个输入。

(4) 与。若 c_1 和 c_2 都是1，则 e_1 为1；否则 e_1 为0。“与”也可有任意多个输入。

因果图的4种关系如图5-16所示。

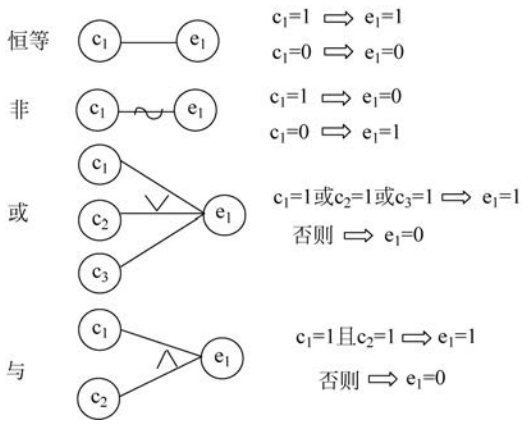


图 5-16 因果图基本符号

在实际问题中输入状态相互之间、输出状态相互之间可能存在某些依赖关系，称为“约束”。为了表示原因与原因之间，结果与结果之间可能存在的约束条件，在因果图中可以附加一些表示约束条件的符号。对于输入条件的约束有 E、I、O、R 四种约束，对于输出条件的约束只有 M 约束。输入输出约束图形符号如图 5-17 所示。

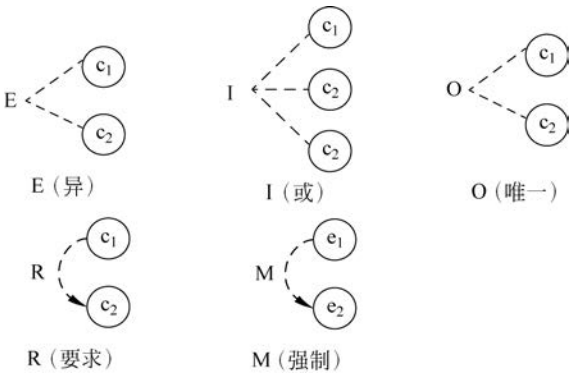


图 5-17 输入输出约束图形符号

为便于理解，这里设 c_1、c_2 和 c_3 表示不同的输入条件。

E(异)：表示 c_1、c_2 中至多有一个可能为 1，即 c_1 和 c_2 不能同时为 1。

I(或)：表示 c_1、c_2、c_3 中至少有一个是 1，即 c_1、c_2、c_3 不能同时为 0。

O(唯一)：表示 c_1、c_2 中必须有一个且仅有一个为 1。

R(要求)：表示 c_1 是 1 时，c_2 必须是 1，即不可能 c_1 是 1 时 c_2 是 0。

M(强制)：表示如果结果 e_1 是 1 时，则结果 e_2 强制为 0。

5.5.2 因果图测试法

因果图可以很清晰地描述各输入条件和输出结果的逻辑关系。如果在测试时必须考虑输入条件的各种组合，就可以利用因果图。因果图最终生成的是判定表。采用因果图设计测试用例的步骤如下。

(1) 分析软件规格说明描述中哪些是原因，哪些是结果。其中，原因常常是输入条件或是输入条件的等价类；结果常常是输出条件。然后给每个原因和结果赋予一个标识符。并且把原因和结果分别画出来，原因放在左边一列，结果放在右边一列。

(2) 分析软件规格说明描述中的语义，找出原因与结果之间，原因与原因之间对应的是什么关系。根据这些关系，将其表示成连接各个原因与各个结果的“因果图”。

(3) 由于语法或环境限制，有些原因与原因之间，原因与结果之间的组合情况不可能出现。为表明这些特殊情况，在因果图上用一些记号标明约束或限制条件。

(4) 把因果图转换成判定表。首先将因果图中的各原因作为判定表的条件项，因果图的各结果作为判定表的动作项；然后给每个原因分别取“真”和“假”两种状态，一般用“0”和“1”表示；最后根据各条件项的取值和因果图中表示的原因和结果之间的逻辑关系，确定相应的动作项的值，完成判定表的填写。

(5) 把判定表的每一列拿出来作为依据，设计测试用例。

下面通过举例来说明如何用因果图进行测试。

例 5-5　软件规格说明书。

第一列字符必须是 A 或 B，第二列字符必须是一个数字，在此情况下进行文件的修改，但如果第一列字符不正确，则给出信息 L，如果第二列字符不是数字，则给出信息 M。

(1) 根据说明书分析出原因和结果。

原因如下：

C1：第一列字符是 A；

C2：第一列字符是 B；

C3：第二列字符是一数字。

结果如下：

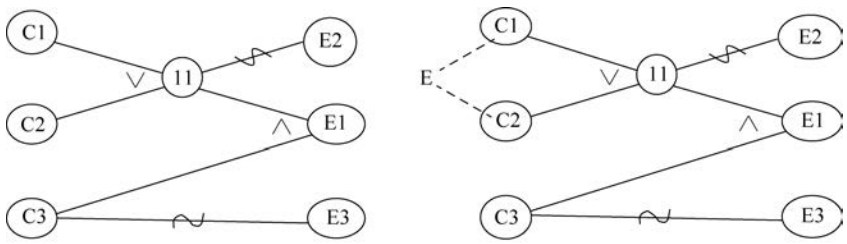
E1：修改文件；

E2：给出信息 L；

E3：给出信息 M。

(2) 绘制因果图。根据原因和结果绘制因果图。把原因和结果用前面的逻辑符号连接起来，画出因果图，如图 5-18(a)所示。

考虑到原因 1 和原因 2 不可能同时为 1，因此在因果图上施加 E 约束。具有约束的因果图如图 5-18(b)所示。



(a) 软件规格说明书因果图　　(b) 有约束的因果图

图 5-18　因果图

注：11 是中间节点

根据因果图所建立的判定表，如表 5-12 所示。

表 5-12 软件规格说明书的判定表

		1	2	3	4	5	6	7	8
条件	C1	1	1	1	1	0	0	0	0
	C2	1	1	0	0	1	1	0	0
	C3	1	0	1	0	1	0	1	0
	11	—	—	1	1	1	1	0	0
动作	E1	/	/	√	0	√	0	0	0
	E2	/	/	0	0	0	0	√	√
	E3	/	/	0	√	0	√	0	√

注意，表 5-12 中规则 1 和规则 2 的原因 1 和原因 2 同时为 1，这是不可能出现的，故应排除这两种情况。因此只需针对第 3～8 列设计测试用例，如表 5-13 所示。

表 5-13 软件规格说明书的测试用例

测试用例	输入数据 a	预期输出
1	A3	修改文件
2	AM	给出信息 M
3	B5	修改文件
4	B*	给出信息 M
5	F2	给出信息 L
6	TX	给出信息 L 和 M

例 5-6 电力收费。

某电力公司有 A、B、C、D 四类收费标准，并做出如下规定。

居民用电 ＜100 度/月，按 A 类收费；≥100 度/月，按 B 类收费。

动力用电 ＜10000 度/月，非高峰，按 B 类收费；≥10000 度/月，非高峰，按 C 类收费；＜10000 度/月，高峰，按 C 类收费；≥10000 度/月，高峰，按 D 类收费。

使用因果图法设计测试用例的步骤和过程如下。

(1) 列出原因和结果。

原因如下：

1——居民用电；

2——动力用电；

3——＜100 度/月，3'——≥100 度/月；

4——非高峰，4'——高峰；

5——≥10000 度/月，5'——＜10000 度/月。

结果如下：

A——按 A 类收费；

B——按 B 类收费；

C——按 C 类收费；

D——按 D 类收费。

（2）用因果图表明输入和输出间的逻辑关系，如图 5-19 所示。

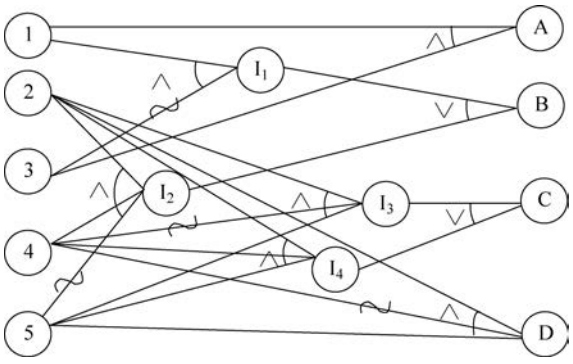


图 5-19 电力收费问题的因果图

注：图中 I1、I2、I3、I4 是中间节点。

（3）把因果图转换为判定表，如表 5-14 所示。

表 5-14 电力收费问题的判定表

组合条件		1	2	3	4	5	6
条件（原因）	1	1	1	0	0	0	0
	2	0	0	1	1	1	1
	3	1	0	—	—	—	—
	4	—	—	1	0	1	0
	5	—	—	0	0	1	1
动作（结果）	A	1	0	0	0	0	0
	B	0	1	1	0	0	0
	C	0	0	0	1	1	0
	D	0	0	0	0	0	1

注：表中“—”表示本条规则与此条件无关，或不适用。

（4）把判定表的每一列写成一个测试用例。

在选取测试用例的时候，可以选择等价类区间中的任何一个数，但为了检查边界处的情况，可以选取等价区间中边界上的值，或其他比较特殊的值，以便提高测试效率。根据判定表设计的测试用例如表 5-15 所示。

表 5-15 电力问题的测试用例

测试用例编号	输入数据	预期结果	覆盖组合条件
1	居民用电，99 度/月	A 类收费	1
2	居民用电，101 度/月	B 类收费	2
3	动力电，非高峰，9000 度/月	B 类收费	3
4	动力电，非高峰，1.01 万度/月	C 类收费	4
5	动力电，高峰，0.98 万度/月	C 类收费	5
6	动力电，高峰，1.1 万度/月	D 类收费	6

5.6 其他黑盒测试方法

除了前面介绍的几种常用的黑盒测试方法外，还有一些黑盒的测试方法，如正交试验法、场景测试法、错误推测法、随机测试法等。

5.6.1 正交试验法

在实际的软件项目中，作为输入条件的原因可能会非常多，如果用因果图进行分析，其因果图可能很庞大，由因果图得到的测试用例数量将达到惊人的程度，这给软件测试工作带来了沉重负担。为了有效、合理地减少测试的费用，可以利用实际生活中行之有效的正交试验设计法来进行测试用例的设计。正交试验测试策略提供了一种能对所有变量对的组合进行典型覆盖（均匀分布）的方法。这种技术对软件组件的集成测试尤其有用，特别是面向对象的系统。

正交试验设计法（Orthogonal Experimental Design）是研究与处理多因素多水平实验的一种科学方法。正交试验设计法是根据正交性从全面试验中挑选出适量的、有代表性的点进行试验，这些点具备了“均匀分散，整齐可比”的特点。利用该方法可以使所有因子和水平在实验中均匀地分布与搭配，均匀规律地变化。在正交试验中，生成正交表是非常重要的。

1. 正交表概念

正交表是运用组合数学理论在正交拉丁名的基础上构造的一种规格化的表格，其符号为 Ln(ji)，其中：

L——正交表的符号；

n——正交表的次数（Runs），即正交表行数；

j——正交表的水平数（Levels），任何单个因素能够取得的值的最大个数；

i——正交表的因素数（Factors），正交表中列的个数，它直接对应到用这种技术设计测试用例时的变量的最大个数。

各列水平数均相同的正交表，称单一水平正交表。各列水平均为 2 的常用正交表有 $L_4(2^3)$、$L_8(2^7)$、$L_{12}(2^{11})$、$L_{16}(2^{15})$、$L_{20}(2^{19})$、$L_{32}(2^{31})$。各列水平数均为 3 的常用正交表有 $L_9(3^4)$、$L_{27}(3^{13})$。各列水平数均为 4 的常用正交表有 $L_{16}(4^5)$。

各列水平数不相同的正交表，称为混合水平正交表。例如 $L_8(4^1\times2^4)$，此混合水平正交表含有 1 个 4 水平列，4 个 2 水平列，共有 1+4=5 列。$L_8(4^1\times2^4)$常简写为 $L_8(4\times2^4)$。

正交表具有以下两个重要的特性。

（1）整齐可比性。在同一张正交表中，每个因素的每个水平出现的次数是完全相同的。由于在试验中每个因素的每个水平与其他因素的每个水平参与试验的概率是完全相同的，这就保证在各个水平中最大程度地排除了其他因素水平的干扰。

（2）均衡分散性。在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对）是完全相同的。这样就保证了试验条件均衡地分散在因素水平的完全组合之中，因而

具有很强的代表性。

2. 选择正交表的基本原则

选择正交表时，一般都是先确定试验的因素、水平和交互作用，后选择适用的 L 表。在确定因素的水平数时，主要因素宜多安排几个水平，次要因素可少安排几个水平。

(1) 先看水平数。若各因素全是 2 水平，就选用 $L(2^*)$表；若各因素全是 3 水平，就选 $L(3^*)$表。若各因素的水平数不相同，就选择适用的混合水平表。

(2) 每一个交互作用在正交表中应占一列或二列。要看所选的正交表是否足够大，能否容纳得下所考虑的因素和交互作用。

(3) 看试验精度的要求。若要求高，则宜取实验次数多的 L 表。若试验费用很昂贵，或试验的经费很有限，或人力和时间都比较紧张，则不宜选实验次数太多的 L 表。

(4) 按原来考虑的因素、水平和交互作用去选择正交表，若无正好适用的正交表可选，简便且可行的办法是适当修改原定的水平数。

3. 用正交表设计测试用例

对软件专业人员来说，测试用例的选择是既有趣又困难的选择。正交表提供了一个选择测试集的方法，可保证对所有被选变量的成对组合，生成一个有效且精简的测试集。这个测试集包括最少的测试用例又可测试所有变量的所有组合，而且生成所有的成对组合是均匀分布的测试集。

用正交表设计测试用例，可按下列步骤进行：

(1) 确定测试中有多少个相互独立的变量，映射到表中的因素数；

(2) 确定每个变量可以取值的最大个数，映射到表中的水平数；

(3) 选择一个次数最少的最适合的正交表。一个最合适的正交表是至少满足第一步说明的因素数且至少满足第二步说明的水平数；

(4) 把变量的值映射到正交表中；

(5) 把每一行的各因素水平的组合作为一个测试用例；

(6) 再增加一些没有在表中出现，但认为可疑的测试用例。

利用正交试验设计方法设计测试用例，可控制生成的测试用例数量，覆盖率高且测试效率高。

5.6.2 场景测试法

1. 场景定义

场景技术在软件开发中可以用来捕获需求和系统的功能，是软件体系结构建模的主要依据，并可用来指导测试用例生成。本质上，场景从用户的角度描述系统的运行行为，反映了系统的期望运行方式。场景是由一系列相关的活动组成的，而且场景中的活动还可以由一系列的场景构成。

现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这一系列的过程利用场景法可以清晰地

描述清楚。这种在软件设计方面的思想也可以引入软件测试中，可以比较生动地描绘出事件触发时的情景，有利于设计测试用例，同时使测试用例更容易理解和执行。通过运用场景来描述系统的功能点或业务流程，从而提高测试效果。

场景一般包含基本流和备用流。从一个流程开始，通过描述经过的路径来确定过程，经过遍历所有的基本流和备用流来完成整个场景。

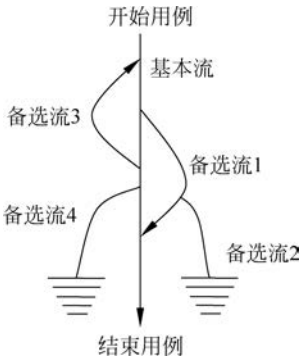


图 5-20　基本流和备选流

对于基本流和备选流的理解，可参考图 5-20。图中经过用例的每条路径都反映了基本流和备选流，都用箭头来表示。中间的直线表示基本流，是经过用例的最简单的路径。备选流用曲线表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中；也可能起源于另一个备选流，或者终止用例而不再重新加入某个流。

根据图 5-20 中每条经过用例的可能路径，可以确定不同的用例场景。从基本流开始，再将基本流和备选流结合起来，可以确定以下用例场景：

场景 1：基本流；

场景 2：基本流、备选流 1；

场景 3：基本流、备选流 1、备选流 2；

场景 4：基本流、备选流 3；

场景 5：基本流、备选流 3、备选流 1；

场景 6：基本流、备选流 3、备选流 1、备选流 2；

场景 7：基本流、备选流 4；

场景 8：基本流、备选流 3、备选流 4。

注：为方便起见，场景 5、6 和 8 只描述了备选流 3 指示的循环执行一次的情况。

2. 场景测试步骤

使用场景法设计测试用例的基本设计步骤如下：

(1) 根据说明书或规约，分析出系统或程序功能的基本流及各项备选流；

(2) 根据基本流和各项备选流生成不同的场景；

(3) 对每一个场景生成相应的测试用例；

(4) 对生成的所有测试用例重新复审，去掉多余的测试用例。测试用例确定后，对每一个测试用例确定测试数据。

3. 网站购物案例

某购物网站订购一般过程是：进入购物网站浏览商品，当看中心仪的商品后，单击立即购买，这时网站弹出登录界面，用户登录自己已注册好的账户，确认收货地址和订单信息，然后通过网上银行支付货款，生成订单。

1) 业务流分析

基本流：浏览商品、立即购买、登录账户、确认收货地址、确认订单信息、支付货款、生成订单。

备选流 1：密码错误；

备选流 2：注册账户；

备选流 3：新增或修改收货地址；

备选流 4：修改订单信息；

备选流 5：网银密码错误；

备选流 6：网银余额不足；

备选流 7：退出系统。

以上只列出了最常见的备选流，在用户购物过程中，还会有很多特殊的情况，在此未完全列出。

2）场景设计

根据上面列出的基本流和备选流，可以构建出大量的用户场景。下面列出部分典型的场景，如表 5-16 所示。

表 5-16 网站购物场景

场 景	业 务流
场景 1：成功购物	基本流
场景 2：用户密码错误	基本流＋备选流 1
场景 3：注册新用户	基本流＋备选流 2
场景 4：新增收货地址	基本流＋备选流 3
场景 5：修改订单	基本流＋备选流 4
场景 6：网银密码错误	基本流＋备选流 5
场景 7：网银余额不足	基本流＋备选流 6
场景 8：选择商品后退出系统	基本流＋备选流 7
场景 9：成功购物	基本流＋备选流 3＋备选流 4

3）测试用例设计

对于每一个场景，需要设计测试数据，使系统按场景中的流程执行。针对表 5-16 中设计的场景，进行测试用例设计，如表 5-17 所示。

表 5-17 网站购物测试用例

项目名称	网站购物测试	项目编号	Shoping_Test		
模块名称	网站购物	开发人员	XXX		
测试类型	功能测试	参考信息	需求规格说明书、设计说明书		
优先级	高	用例作者	XXXX	设计日期	XXXX
测试方法	黑盒测试(手工测试)	测试人员	XXXX	测试日期	XXXX
测试对象	网站购物业务功能的测试				
前置条件	用户账户：wangmin，密码：126543lan，网银账号：622848 ********** ，密码：123456，账户余额：300 元				

续表

用例编号	场景	操作描述	输入数据	期望结果	实际结果
Shoping_Test_1	场景 1	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 确认收货地址 5. 确认订单信息 6. 支付货款 7. 生成订单	用户账户：wangmin 密码：126543lan 网银账号：622848 ********** 密码：123456 商品价格：80 元 商品数量：1	购物成功	
Shoping_Test_2	场景 2	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码	用户账户：wangmin 密码：126543	提示密码错误	
Shoping_Test_3	场景 3	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 注册新用户	用户账户：wangyu 密码：yu123456	用户注册成功	
Shoping_Test_4	场景 4	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 新增收货地址，输入新的地址 5. 确认收货地址 6. 确认订单信息 7. 支付货款 8. 生成订单	用户账户：wangmin 密码：126543lan 新地址：四川省绵阳市花园小区 网银账号：622848 ********** 密码：123456 商品价格：60 元 商品数量：1	购物成功	
Shoping_Test_5	场景 5	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 确认收货地址 5. 修改订单信息(修改购买商品的数量)，然后确认订单信息 6. 支付货款 7. 生成订单	用户账户：wangmin 密码：126543lan 网银账号：622848 ********** 密码：123456 商品价格：60 元 商品数量：4	购物成功	
Shoping_Test_6	场景 6	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 确认收货地址 5. 确认订单信息 6. 支付货款，但密码错误	用户账户：wangmin 密码：126543lan 网银账号：622848 ********** 密码：111111 商品价格：60 元 商品数量：1	提示密码错误	
Shoping_Test_7	场景 7	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 确认收货地址 5. 确认订单信息 6. 支付货款	用户账户：wangmin 密码：126543lan 网银账号：622848 ********** 密码：123456 商品价格：360 元 商品数量：1	提示余额不足	

续表

用例编号	场景	操作描述	输入数据	期望结果	实际结果
Shoping_Test_8	场景 8	1. 浏览商品并确定要购买的商品 2. 单击“加入购物车”按钮 3. 退出本网站		商品加入购物车	
Shoping_Test_9	场景 9	1. 浏览商品并确定要购买的商品 2. 单击“立即购买”按钮 3. 输入账户和密码 4. 新增收货地址，然后确认收货地址 5. 修改订单信息(修改购买商品的数量)，然后确认订单信息 6. 支付货款 7. 生成订单	用户账户：wangmin 密码：126543lan 新地址：四川省绵阳市花园小区 网银账号：622848 ********** 密码：123456 商品价格：40 元 商品数量：6	购物成功	

5.6.3 错误推测法

人们可以靠经验和直觉推测程序中可能存在的各种错误，从而有针对性地编写检查这些错误的例子，这就是错误推测法。错误推测法的基本思想是列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据这些特殊情况选择测试用例。

用错误推测法进行测试，首先需罗列出可能的错误或错误倾向，进而形成错误模型；然后设计测试用例以覆盖所有的错误模型。例如，对一个排序的程序进行测试，其可能出错的情况如下：输入表为空的情况；输入表中只有一个数字；输入表中所有的数字都具有相同的值；输入表已经排好序等。

错误推测法是一种简单易行的黑盒法，但由于该方法有较大的随意性，主要依赖于测试者的经验，因此通常作为一种辅助的黑盒测试方法。

5.7 本章小结

黑盒测试是通过程序的输入和输出来检测每个功能是否都能正常使用。从理论上讲，黑盒测试只有采用穷举输入测试，把所有可能的输入都作为测试情况考虑，才能查出程序中所有的错误。实际上测试情况有无穷多个，不仅要测试所有有效(合法)的输入，而且还要对所有可能发生的无效(不合法)输入进行测试，因此完全测试是不可能的。我们应该进行有针对性地测试，通过制定测试方案和测试用例指导测试的实施，保证软件测试有组织、按步骤、有计划地进行。

本章介绍了黑盒测试的方法，特别对等价类划分、边界值分析、因果图、基于判定表的测试方法进行介绍，并通过实例展示了各种黑盒测试方法设计测试用例的过程。

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。

边界值分析是通过选择输入或输出的边界设计测试用例。边界值分析法不仅重视输入条件边界，而且有时还必须考虑输出域边界。

因果图方法是从用自然语言书写的程序规格说明的描述中找出因(输入条件)和果(输出或程序状态的改变)之间的关系绘制出因果图，然后通过因果图转换为判定表。

正交试验设计法是使用已经造好的正交表来安排试验并进行数据分析的一种方法，目的是用最少的测试用例达到最高的测试覆盖率。

错误推测设计方法是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。

进行黑盒测试时需要根据被测试对象选择合适的测试方法。Myers 提出了使用各种测试方法的综合策略：

(1) 在任何情况下都必须使用边界值分析方法。经验表明用这种方法设计出测试用例发现程序错误的能力最强。

(2) 必要时用等价类划分方法补充一些测试用例。

(3) 用错误推测法再追加一些测试用例。

(4) 对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度。如果没有达到要求的覆盖标准，应当再补充足够的测试用例。

(5) 如果程序的功能说明中含有输入条件的组合情况，则一开始就可选用因果图法。

思考题

1. 边界值分析方法如何生成测试用例?

2. 如何使用等价类测试技术生成测试用例?

3. 如何结合使用等价类测试技术和边界值分析技术设计测试用例?

4. 请简述利用因果图生成测试用例的基本步骤。

5. 简述使用判定表设计测试用例的一般过程。

6. 某程序对毕业设计成绩进行评定，90～100 分为优秀，75～89 分为良，60～74 分为及格，0～59 分为不及格。请用边界值分析方法设计测试用例。

7. 程序有三个输入变量 month、day、year(month、day 和 year 均为整数值，并且满足 1≤month≤12、1≤day≤31、1800≤ year ≤2050。)，分别作为输入日期的月份、日、年份，通过程序可以输出该输入日期在日历上隔一天(第三天)的日期。例如，输入为 2005 年 11 月 29 日，则该程序的输出为 2005 年 12 月 1 日。请用边界测试法设计测试用例。

8. 某系统注册页面需要输入用户名和密码，其中要求：①用户名只能包含英文字母、下画线、数字，且字符长度为 4 至 12 个字符；②密码为 6 到 10 位，且只能包含字母和数字。如果用户名和密码符合要求，提示注册成功；否则，输出相应的错误信息。请使用弱健壮等价类测试法设计测试用例。

9. 某程序的功能是输入一组整数数据(数据个数不超过 100 个)，使用冒泡排序法进行排序，数据按从小到大的顺序排列。请用等价类方法设计测试用例。

10. 某博客系统的相册功能模块可以上传图片，其中要求：上传图片的格式只能是 gif、

jpg 和 bmp，图片的大小不超过 5MB。如果图片格式和大小符合要求，则上传成功；否则给出相应的错误提示信息。请用弱健壮等价类测试方法设计测试用例。

11. 某程序的功能是实现金额校验。金额范围[0.00,99.99](最多只能有两位小数)。例如，输入为 56.57，输出为 56 元 5 角 7 分；输入为 21，输出为 21 元；输入为 23.4，输出为 23 元 4 角。请用等价类方法设计测试用例。

12. 某程序的功能是根据消费者当日消费总额计算其可享受的优惠金额。具体计算规则如下：优惠金额＝消费总额×优惠折扣＋现金红包，优惠折扣、现金红包和消费总额之间的规则如表 5-18 所示。请用边界值分析方法和基于判定表的方法设计测试用例。

表 5-18 优惠折扣、现金红包和消费总之间的规则

当日消费	优惠折扣	现金红包
消费累计≤500 元	2%	0 元
500 元＜消费累计≤1000 元	5%	100 元
消费累计＞ 1000 元	10%	300 元

13. 某学校评定奖学金的要求是：A 级奖学金，所有学科的平均成绩≥85 分，没有不及格的课程，且至少参加学生科技活动奖一项；B 级奖学金，所有学科的平均成绩≥85 分，且没有不及格的课程。请用基于决策表的测试方法进行测试。要求：绘制决策表，设计测试用例。

14. 某程序的功能是：从键盘输入当月利润，求应发放奖金总数。企业发放的奖金是根据利润提成，利润低于或等于 10 万元时，奖金可提 10%；利润在 10 万元到 50 万元之间，高于 10 万元的部分，奖金可提 5%；50 万元到 100 万元之间时，高于 50 万元的部分，奖金可提 2%；高于 100 万元时，超过 100 万元的部分按 1%提成(例如利润为 55 万元，奖金＝10×10%＋40×5%＋5×2%＝3.1 万元)。请用基于决策表的测试方法设计测试用例。

15. 请用场景测试法测试 QQ 邮件系统发送普通邮件(不带附件)的功能。发送邮件的基本步骤是：输入正确的用户名和密码登录邮件系统，单击“写信”按钮，撰写邮件。写完邮件后，单击“发送”按钮发送邮件。要求：写出基本流和备选流，然后设计场景。

16. 请使用因果图法为三角形问题设计测试用例。