SOFTWARE TESTING

Chapter 5

Black Box and Gray Box Testing

School of Data & Computer Science
Sun Yat-sen University

Approaches & Technologies

OUTLINE

- 5.1 黑盒测试
  - 黑盒测试概述
  - 等价类划分法
  - 边界值分析法
  - 错误推测法
  - 随机测试法
  - 因果图法
- 5.2 灰盒测试

黑盒测试概述

黑盒测试的概念

- 黑盒测试把测试对象当作看不见内部细节的“黑盒”，在完全不考虑被测程序内部结构和处理过程的情况下，仅仅依据程序功能的需求规格设计测试用例，并推断测试结果的正确性。
  - 黑盒测试也称为功能测试或数据驱动测试。黑盒测试要求导出执行被测程序所有功能需求的输入条件集，生成测试用例集，实现功能覆盖。
    - 功能覆盖主要是需求覆盖，即通过设计一定的测试用例，对每个功能需求点进行测试。
    - 根据软件产品需求规格说明中的功能设计规格进行测试，以证实每个实现了的功能是否符合规格要求。
- 比较：白盒测试按照程序内部逻辑结构和编码结构来设计测试用例并完成测试，因此又称为结构测试或逻辑驱动测试。

黑盒测试概述

黑盒测试的概念 (续)

- 黑盒测试在被测软件的接口处进行。
  - 黑盒测试着眼于被测程序的外部结构，而不考虑其内部逻辑，主要针对软件界面和软件功能进行测试。
  - 黑盒测试站在软件使用者的角度，从输入数据与输出数据的对应关系出发进行测试。
- 黑盒测试需要解决的问题：
  - 如何测试功能的有效性？
  - 如何测试系统的行为和性能？
  - 如何测试系统能够承受的数据速率和数据量？
  - 何种类型的输入会产生好的测试用例？
  - 系统是否对特定的输入值特别敏感？
  - 如何分隔数据类的边界？

黑盒测试概述

- 黑盒测试的局限
  - 黑盒测试无法发现被测程序的外部特性设计或需求规格说明的规定有误。
  - 黑盒测试不能替代白盒测试，而是用来发现白盒测试以外的其他类型的错误，比如：
    - 功能错误或遗漏
    - 接口错误或界面错误
    - 数据结构或外部数据库访问错误
    - 性能错误
    - 初始化和中止错误

黑盒测试概述

黑盒测试的局限 (续)

- 黑盒测试需要在所有可能的输入/输出条件中确定测试用例，检查程序是否都能产生预期输出，以期发现被测程序中的错误。
  - 一般情况下难以实现。穷举测试数量太大，不仅要测试所有合法的输入，而且还要对那些不合法但是可能的输入进行测试。
    - 假设一个程序 P 有输入量 X 和 Y 及输出量 Z。在字长为32位计算机上运行。若 X、Y 取整数，按黑盒方法进行穷举测试，其测试数据组数量可能为 232×232=264。
    - 如果测试一组数据需要1毫秒，一年工作 365×24 小时，完成 P 的所有测试大约需要5亿年。

黑盒测试概述

- 黑盒测试方法
  - 黑盒测试需要确定一些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“任何情况”。
  - 常用的黑盒测试方法：
    - 等价类划分法
    - 边界值分析法
    - 猜错法
    - 随机数法
    - 因果图方法
    - 。。。
  - 这些测试方法从更广泛的角度进行黑盒测试，每种方法都力图能涵盖更多的“任何情况”，但又各有长处。
  - 综合使用这些方法，可以得到较好的测试用例集。

黑盒测试概述

黑盒测试要求

- 每个被测程序的特性或功能必须被一个测试用例或一个被认可的异常所覆盖。
- 既要考核“程序是否做了应该做的”，还要考察“程序是否没有做不应该做的”。
  - 同时还要考察被测程序在其他一些情况下是否正常，这些情况包括数据类型和数据值的异常等等。
- 需要构造数据类型和数据值的最小集测试。
  - 用一系列真实的数据类型和数据值运行，测试超负荷、饱和及其他“最坏情况”结果。
- 通过假想的数据类型和数据值测试系统排斥非法输入的能力。
- 对影响系统性能的关键模块 (如基本算法) 的模块性能 (如精度、时间、容量等) 进行测试。

黑盒测试概述

黑盒测试要求 (续)

- 黑盒测试设计的测试用例集需要满足以下两个标准：
  - 所设计的测试用例能够减少为达到合理测试所需要设计的测试用例的总数；
  - 所设计的测试用例能够揭示是否存在某些类型的错误，而不是仅仅指出与特定测试有关的错误是否存在。

黑盒测试概述

- 黑盒测试的通过准则
  - 黑盒测试只有测试通过和测试失败两种结果。
  - 在设计和执行测试用例时，首先考虑“通过测试”的用例。
    - 通过测试实际上只是确认被测软件能够做什么，而不去考验其能力如何。
    - 通过测试的目的是在进行破坏性试验之前，先考察被测软件是否实现了基本功能。
    - 通过测试的用例尽量做到简单、直观。
  - 在确信被测软件能够正常运行之后，再采取各种手段 (例如压力测试) 找出其它软件缺陷。

黑盒测试概述

- 黑盒测试的测试用例开发
  - 黑盒测试与被测程序如何具体实现无关。即使程序的实现发生变化，黑盒测试用例仍应该可用 (可重用性，面向回归测试)。
    - 黑盒测试用例开发可以与软件开发同时进行，从而节省总体开发时间。通过软件用例可以设计出大部分的黑盒测试用例。
  - 黑盒测试用例存在的一些问题：
    - 测试用例数量较大；
    - 测试用例可能产生很多冗余；
    - 功能性测试的覆盖范围难以达到完全覆盖。

等价类划分法

- 等价类划分是一种典型的黑盒测试方法。
  - 等价类划分方法完全不考虑被测程序的内部结构，只依据程序的规格说明来设计测试用例。
  - 等价类划分方法把所有可能的输入数据 (即程序的输入域) 划分成若干部分，然后从每一部分中选取少数有代表性的数据做为测试用例。
  - 使用等价类划分方法设计测试用例要经历划分等价类 (列出等价类表) 和确立测试用例两个步骤。

等价类划分法

- 等价类的划分
  - 一个输入等价类是指程序输入域的某个子集，在该子集中，各个输入数据对于揭露程序中的错误是等效的。测试某一个等价类的代表值就等同于对这个等价类的其它值的测试。
  - 等价类的划分有两种不同的情况：
    - 有效等价类：对于程序规格说明来说是合理的、有意义的输入数据构成的集合。
    - 无效等价类：对于程序规格说明来说是不合理的、无意义的输入数据构成的集合。
  - 设计测试用例时，通常要求同时考虑有效等价类和无效等价类的用例设计 (即健壮性测试)。

等价类划分法

- 划分等价类的原则

(1) 如果输入条件规定了输入数据的取值范围或者值的个数，则可以确立一个有效等价类和两个无效等价类。

- 例：在程序的规格说明中，输入条件规定 “... 项数可以从1到999 ...”，则有效等价类是 “1≤项数≤ 999”，两个无效等价类是 “项数<1”和“项数>999”。

(2) 如果输入条件规定了输入数据的集合，或者是规定了“必须如何”的条件，则可以确立一个有效等价类和一个无效等价类。

- 例：在 Pascal 语言中对变量标识符规定为“以字母打头的 ... 串”。那么所有以字母打头的串构成有效等价类，而不在此集合内的串 (不以字母打头的串) 归于无效等价类。

等价类划分法

- 划分等价类的原则 (续)

(3) 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类。

(4) 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理，这时可为每一个输入值确立一个有效等价类，此外针对这组值确立一个无效等价类，它是所有不允许的输入值的集合。

- 例：初等几何中对等腰三角形、平行四边形、梯形和圆形四类规则平面图形的面积分别采用相应的面积公式进行计算。
  - 可以确定4个有效等价类为等腰三角形、平行四边形、梯形和圆形，而由所有非上述四类图形的其它平面图形的集合构成一个无效等价类。

等价类划分法

划分等价类的原则 (续)

(5) 如果规定了输入数据必须遵守的规则，则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)。

- 例：Pascal 语言规定“一个语句必须以分号 ‘;’ 结束”。这时，可以确定一个有效等价类“以 ‘;’ 结束”，若干个无效等价类“以 ‘:’ 结束”、“以 ‘,’ 结束”、“以 ‘ ’ 结束”、“以 LF 结束” 等等。

等价类划分法

- 测试用例的确立
  - 完成等价类划分之后，建立等价类表，列出所有划分得到的等价类，并按以下原则选择测试用例：
    - 为每一个等价类规定一个唯一编号；
    - 设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类。重复这一步，直到所有的有效等价类都被覆盖为止；
    - 设计一个新的测试用例，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。

等价类划分法

测试用例的确立 (续)

例：在某 Pascal 语言版本中对变量标识符的规定：

(1) 标识符是由字母开头，后跟字母或数字的任意字符组合构成；构成标识符的有效字符数为 1 - 8 个；标识符的字符总数不能超过80个；

(2) 标识符必须先说明，再使用；

(3) 在同一说明语句中，至少必须有一个标识符。

- 有一个程序片段用于检查上述合规性。用等价类划分法设计该程序片段的黑盒测试用例。

等价类划分法

测试用例的确立 (续)

例：(续)

- 划分得到的15个等价类及编号如下：

输入条件	有效等价类	无效等价类
标识符个数	1个（1），多个（2）	0个（3）
标识符字符数	1～8个（4）	0个（5），＞8个（6），＞80个（7）
标识符组成	字母（8），数字（9）	非字母数字字符（10），保留字（11）
第一个字符	字母（12）	非字母（13）
标识符使用	先说明后使用（14）	未说明已使用（15）

等价类划分法

测试用例的确立 (续)

例：(续) 下面选取9个测试用例，覆盖所有的15个等价类。

```
① VAR x, T1234567: REAL;
   BEGIN  x := 3.414;
          T1234567 := 2.732;
                ……
```
(1), (2), (4), (8), (9), (12), (14)

② `VAR : REAL;` (3)

③ `VAR x, : REAL;` (5)

④ `VAR T12345678: REAL;` (6)

⑤ `VAR T12345 …… : REAL;` (7)
多于80个字符

⑥ `VAR T$: CHAR;` (10)

⑦ `VAR GOTO: INTEGER;` (11)

⑧ `VAR 2T: REAL;` (13)

⑨
```
VAR PAR: REAL;
   BEGIN  ……
     PAP := SIN (3.14 * 0.8) / 6;
```
(15)

等价类划分法

测试用例的确立 (续)

例：(续) 下面选取9个测试用例，覆盖所有的15个等价类。

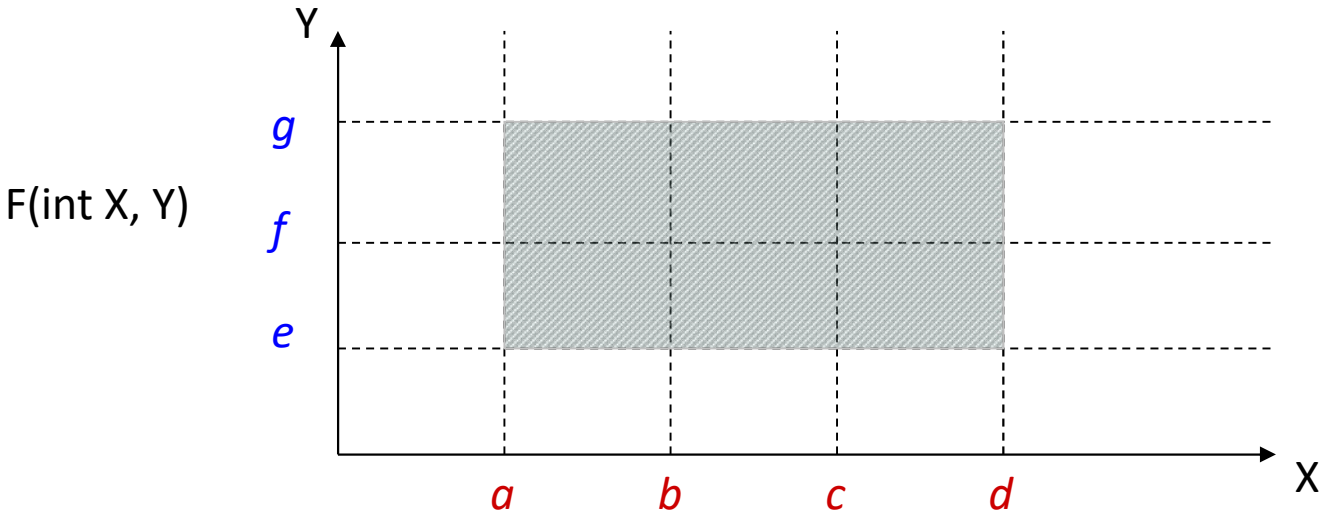
① VAR x, T1234567: REAL; BEGIN x := 3.414; T1234567 := 2.732; ……	(1), (2), (4), (8), (9), (12), (14)
② VAR : REAL;	(3)
③ VAR x, : REAL;	(5)
④ VAR T12345678: REAL;	(6)
⑤ VAR T12345 …… : REAL; 多于80个字符	(7)
⑥ VAR T$: CHAR;	(10)
⑦ VAR GOTO: INTEGER;	(11)
⑧ VAR 2T: REAL;	(13)
⑨ VAR PAR: REAL; BEGIN …… PAP := SIN (3.14 * 0.8) / 6;	(15)

用例对应的等价类编号

- 等价类划分法
  - 弱/强、一般/健壮的等价类测试分类
    - 弱 (weak) 等价类测试
      - 针对单缺陷的等价类测试用例设计。
      - 单缺陷：在同一输入条件下失效大概率由单个缺陷引起。
    - 强 (strong) 等价类测试
      - 针对多缺陷的等价类测试用例设计。
    - 一般 (normal) 等价类测试
      - 只覆盖有效等价类的测试用例设计。
    - 健壮 (robust) 等价类测试
      - 同时覆盖有效等价类和无效等价类的测试用例设计。
      - 健壮性：在异常情况下软件还能正常运行的能力。健壮性包括容错能力和异常恢复能力。容错性测试通常构造一些不合理的输入来诱导软件出错。

等价类划分法

- 弱/强、一般/健壮的等价类测试分类 (续)
  - 设被测程序实现了函数 F(int X, Y)，输入变量 X, Y 拥有以下有效边界以及边界内的区间：
    - $a \le X \le d$，区间为 $[a, b), [b, c), [c, d]$
    - $e \le Y \le g$，区间为 $[e, f), [f, g]$

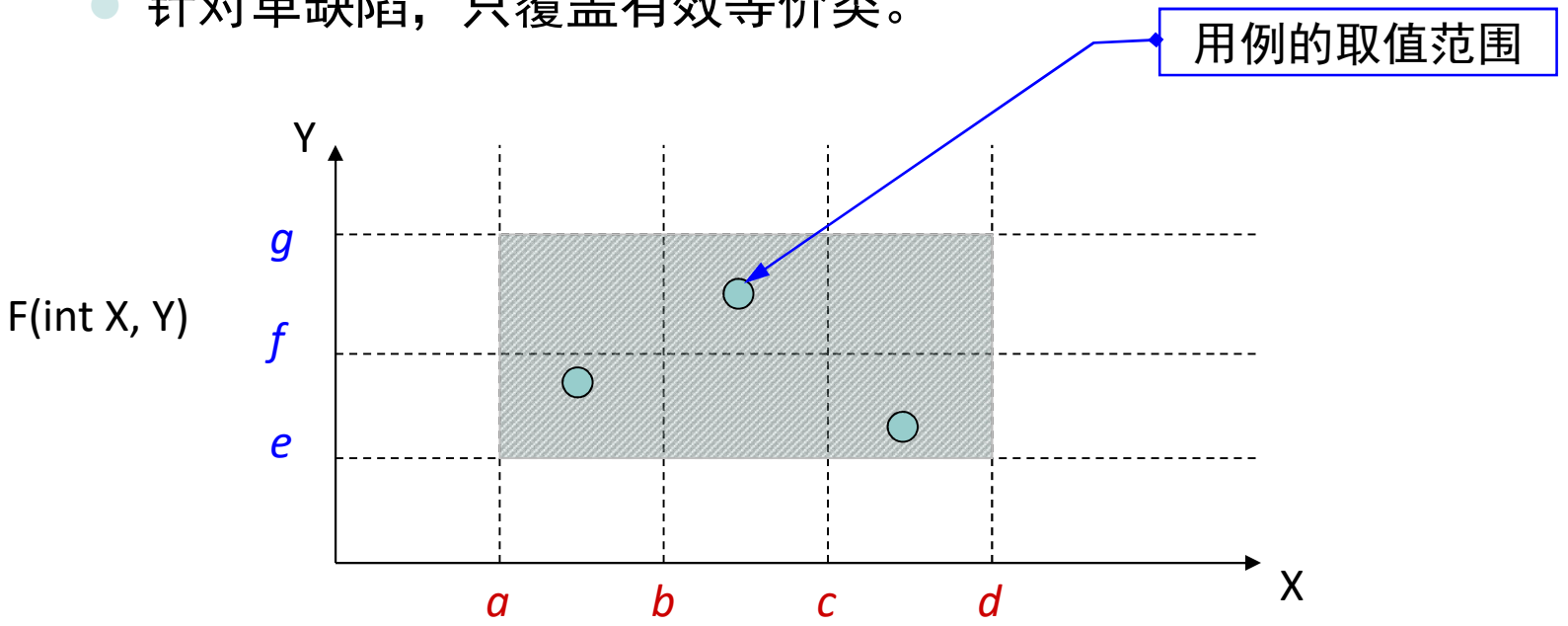


等价类划分法

弱/强、一般/健壮的等价类测试分类 (续)

(1) 弱一般等价类测试用例覆盖

- 针对单缺陷，只覆盖有效等价类。



X 有效等价类 [*a*, *b*), [*b*, *c*), [*c*, *d*]，无效等价类 X<*a*, X>*d*

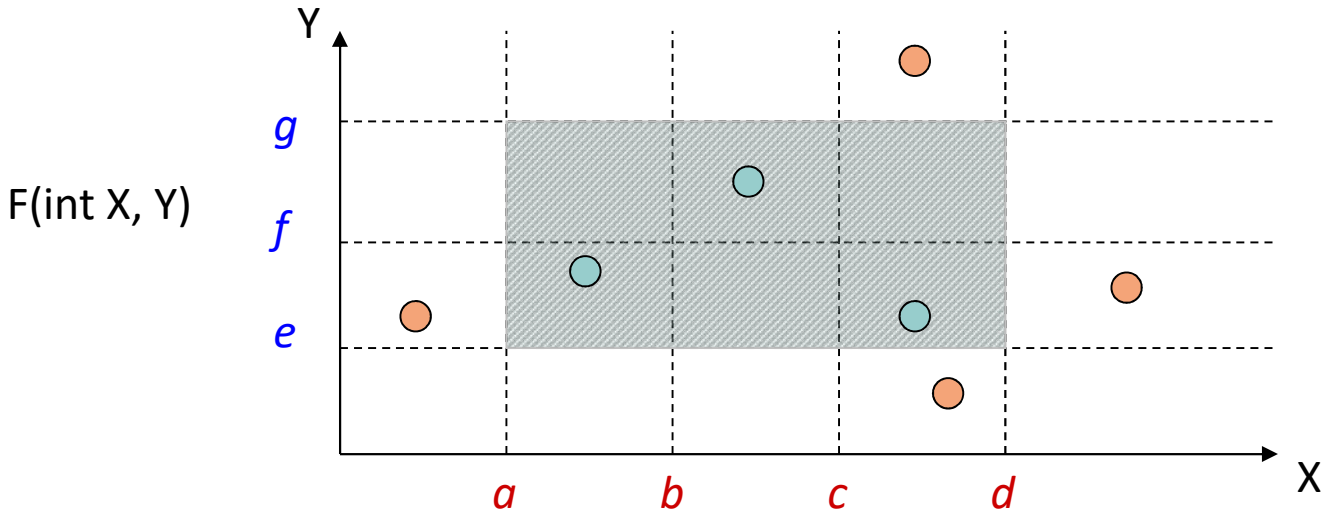
Y 有效等价类 [*e*, *f*), [*f*, *g*]，无效等价类 Y<*e*, Y>*g*

等价类划分法

弱/强、一般/健壮的等价类测试分类 (续)

(2) 弱健壮等价类测试用例覆盖

- 针对单缺陷，覆盖有效等价类和无效等价类



X 有效等价类 [*a*, *b*), [*b*, *c*), [*c*, *d*]，无效等价类 X<*a*, X>*d*

Y 有效等价类 [*e*, *f*), [*f*, *g*]，无效等价类 Y<*e*, Y>*g*

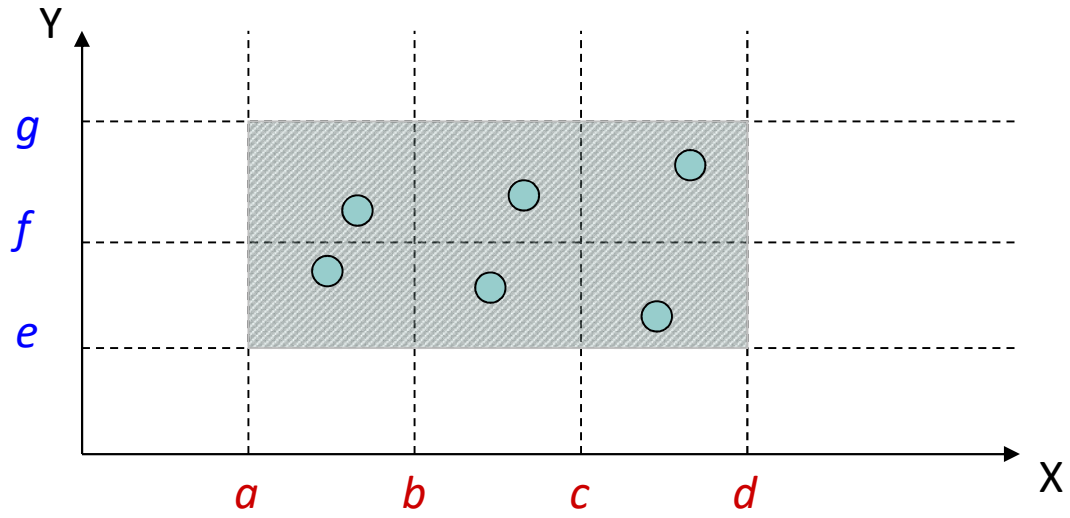
等价类划分法

弱/强、一般/健壮的等价类测试分类 (续)

(3) 强一般等价类测试用例覆盖

- 针对多缺陷，只覆盖有效等价类

F(int X, Y)



X 有效等价类 [*a*, *b*), [*b*, *c*), [*c*, *d*]，无效等价类 X<*a*, X>*d*

Y 有效等价类 [*e*, *f*), [*f*, *g*]，无效等价类 Y<*e*, Y>*g*

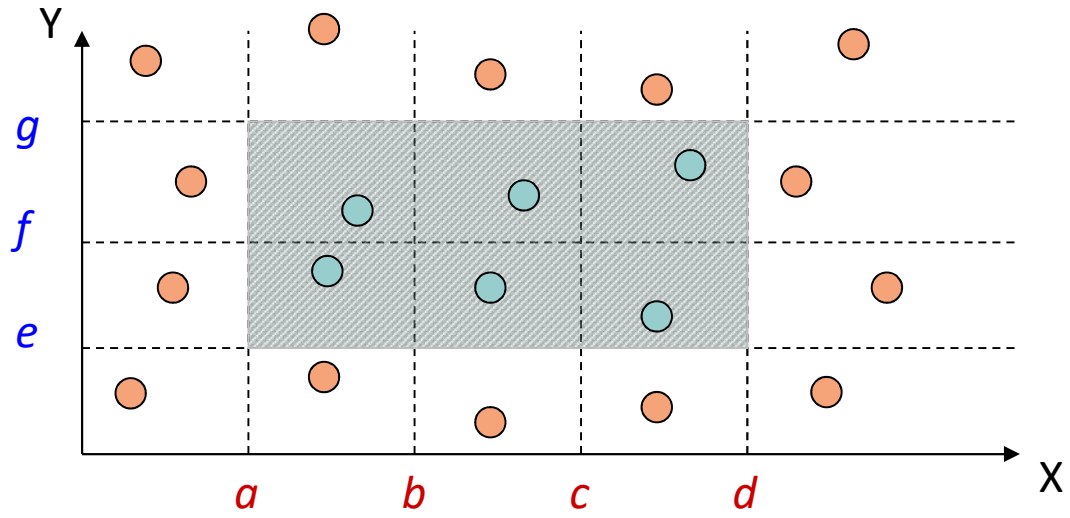
等价类划分法

弱/强、一般/健壮的等价类测试分类 (续)

(4) 强健壮等价类测试用例覆盖

- 针对多缺陷，覆盖有效等价类和无效等价类

F(int X, Y)



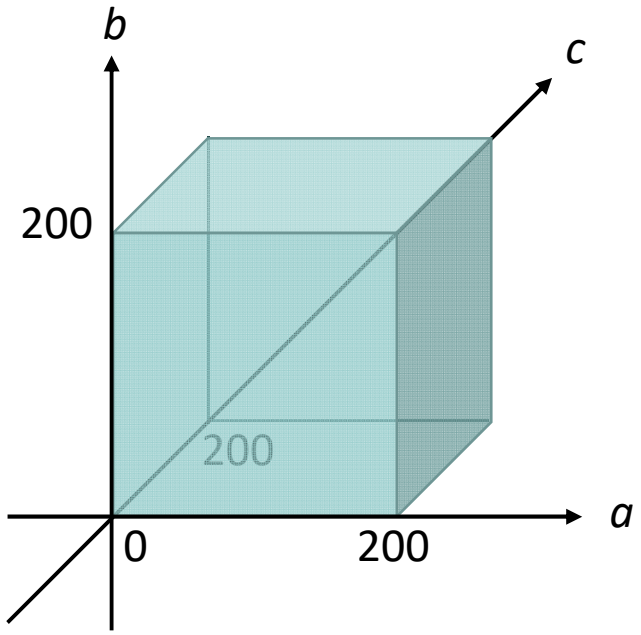
X 有效等价类 [*a*, *b*), [*b*, *c*), [*c*, *d*]，无效等价类 X<*a*, X>*d*

Y 有效等价类 [*e*, *f*), [*f*, *g*]，无效等价类 Y<*e*, Y>*g*

等价类划分法

Equivalence Class test Cases for the triangle problem

Let $a, b, c \in (0, 200]$ be input as the lengths of three sides of a triangle, we note that four possible outputs can occur: NotATriangle (非三角形), Scalene (不等边三角形), Isosceles (等腰三角形), and Equilateral (等边三角形).



等价类划分法

Equivalence Class test Cases for the triangle problem

- Let $a, b, c \in (0, 200]$ be input as the lengths of three sides of a triangle, we note that four possible outputs can occur: NotATriangle (非三角形), Scalene (不等边三角形), Isosceles (等腰三角形), and Equilateral (等边三角形).
- We can use these to identify output (range) equivalence classes as follows.
  - R1 = {<*a*, *b*, *c*>: the triangle with sides *a*, *b*, and *c* is equilateral}
  - R2 = {<*a*, *b*, *c*>: the triangle with sides *a*, *b*, and *c* is isosceles}
  - R3 = {<*a*, *b*, *c*>: the triangle with sides *a*, *b*, and *c* is scalene}
  - R4 = {<*a*, *b*, *c*>: sides *a*, *b*, and *c* do not form a triangle}

等价类划分法

Equivalence Class test Cases for the triangle problem

- Four weak normal equivalence class test cases, chosen arbitrarily from each class are as follows:

Test Case	*a*	*b*	*c*	*Expected Output*
WN1	5	5	5	Equilateral
WN2	2	2	3	Isosceles
WN3	3	4	5	Scalene
WN4	4	1	2	Not a triangle

- Because no valid subintervals of variables *a*, *b*, and *c* exist, the strong normal equivalence class test cases are identical to the weak normal equivalence class test cases.

等价类划分法

Equivalence Class test Cases for the triangle problem

- Considering the invalid values for a, b, and c yields the following additional weak robust equivalence class test cases.

Test Case	*a*	*b*	*c*	*Expected Output*
WR1	-1	5	5	Value of *a* is out of range
WR2	5	-1	5	Value of *b* is out of range
WR3	5	5	-1	Value of *c* is out of range
WR4	201	5	5	Value of *a* is out of range
WR5	5	201	5	Value of *b* is out of range
WR6	5	5	201	Value of *c* is out of range

- The invalid values could be zero, any negative number, or any number greater than 200.

等价类划分法

Equivalence Class test Cases for the triangle problem

Here is one of the eight “corners” of the cube in three-space of the additional strong robust equivalence class test cases:

Test Case	*a*	*b*	*c*	*Expected Output*
SR1	-1	5	5	Value of *a* is out of range
SR2	5	-1	5	Value of *b* is out of range
SR3	5	5	-1	Value of *c* is out of range
SR4	-1	-1	5	Values of *a, b* are out of range
SR5	5	-1	-1	Values of *b, c* are out of range
SR6	-1	5	-1	Values of *a, c* are out of range
SR7	-1	-1	-1	Values of *a, b, c* are out of range

边界值分析法

概述

- 边界值分析法是对等价类划分方法的补充。
- 经验显示，大量的错误发生在输入或输出范围的边界上，而不是在输入范围的内部。如果能够针对各种边界情况设计测试用例，则有可能查出更多的错误。
- 例：在做三角形计算时，需要输入三角形的三个边长：A、B 和 C。 三角形的这三个边长数值应当满足

  $A > 0$、$B > 0$、$C > 0$、$A + B > C$、$A + C > B$、$B + C > A$

  才能构成三角形。
  - 按照 ESTCA (错误敏感测试用例分析) 准则，问题大概率出现在容易被疏忽的边界附近。

边界值分析法

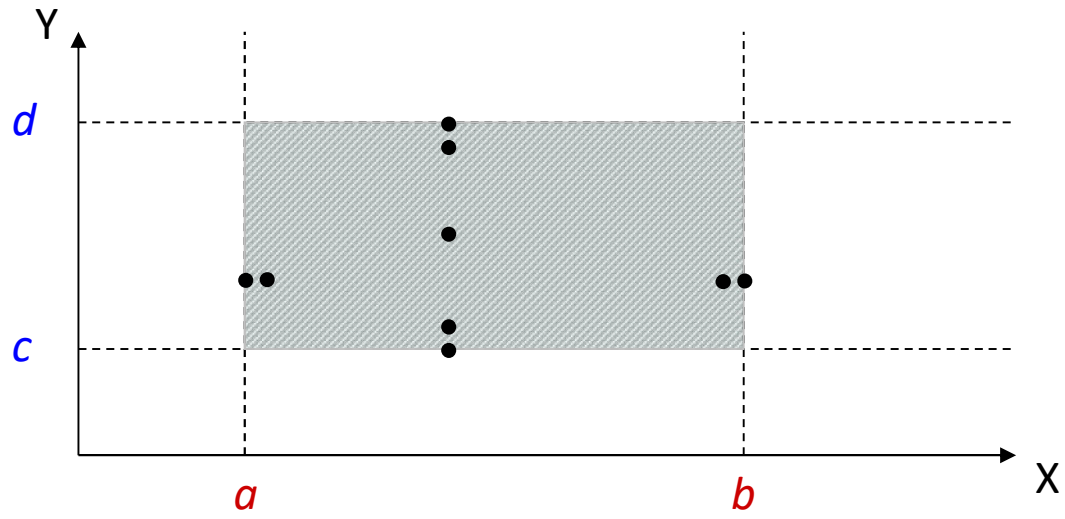
概述 (续)

- 用边界值分析方法设计测试用例，先要确定边界情况。应选取正好等于、刚刚大于、或刚刚小于边界的值做为测试数据，而不是选取等价类中的典型值或任意值做为测试数据。
- 边界值的分析与确定比较复杂，要求测试人员有更多的经验和耐心。
- 设被测程序实现了函数 F(int X, Y)，输入变量 X, Y 拥有以下有效边界：
  - $a \leq X \leq b$
  - $c \leq Y \leq d$

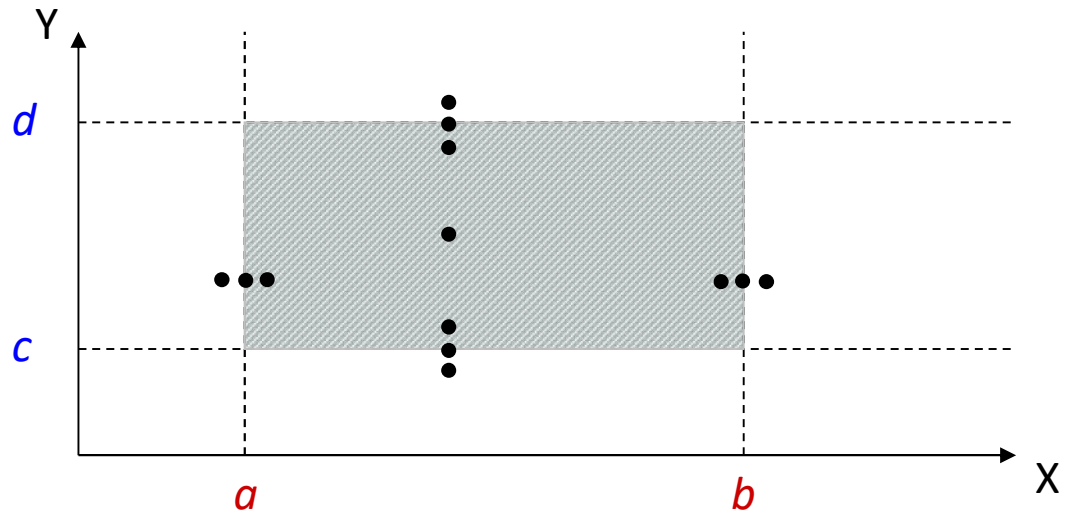
边界值分析法

边界值测试用例

(1) 边界值一般测试用例



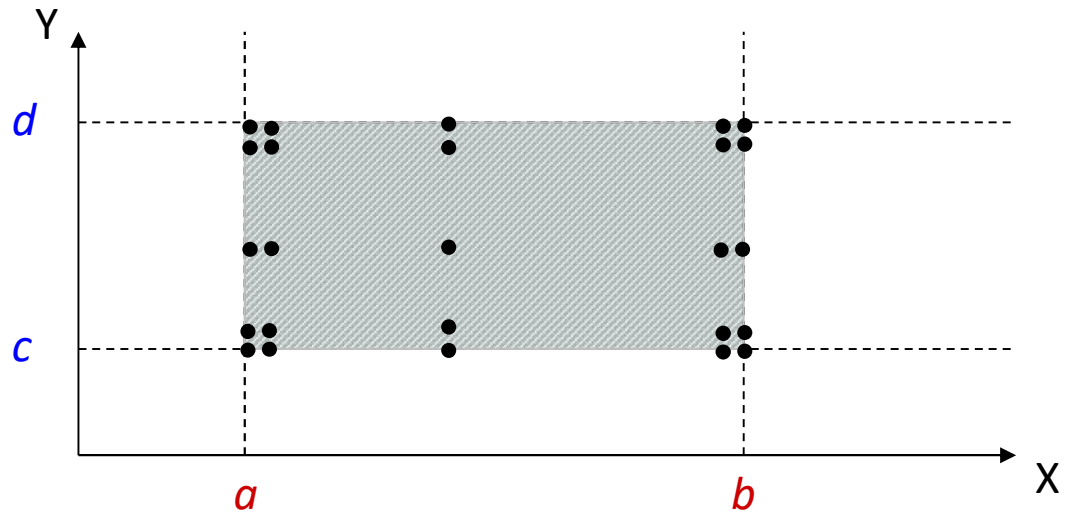
- 边界值分析法
  - 边界值测试用例
    (2) 边界值健壮测试用例



边界值分析法

边界值测试用例

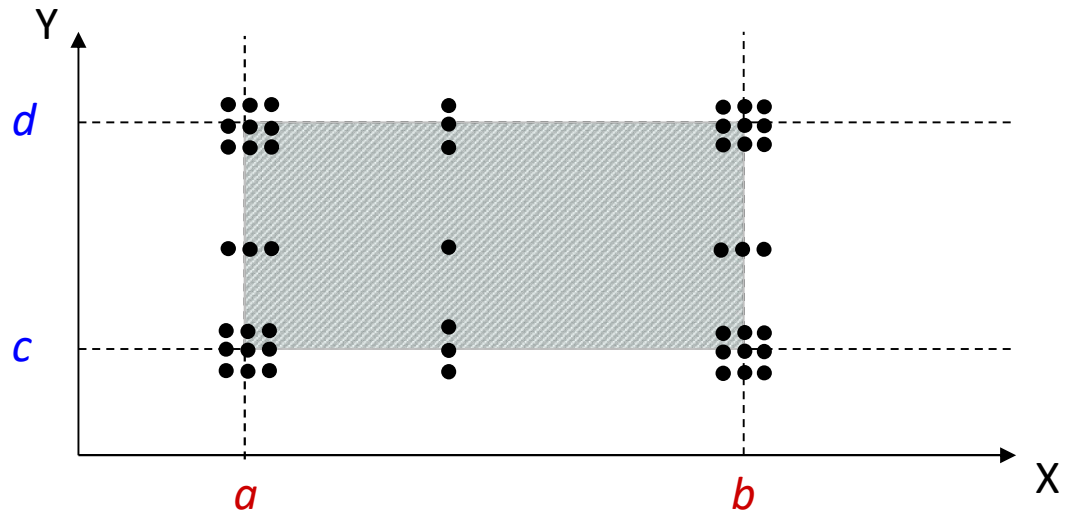
(3) 边界值最坏情况的一般测试用例



边界值分析法

边界值测试用例

(4) 边界值最坏情况的强壮测试用例



边界值分析法

- 等价类划分法与边界值分析法的比较
  - 等价类划分法的测试用例数据在各个等价类允许的取值范围内任意选取，而边界值分析法的测试用例数据必须在边界值附近选取。
  - 一般而言，用边界值分析法设计的测试用例要比等价类划分法更具有代表性，发现错误的能力也更强。

错误推测法

- 概述
  - 错误推测法依靠测试人员的经验和直觉对被测程序中可能存在的各种错误进行推测，从而有针对性地编写测试用例。
- 基本思想
  - 列举程序中所有可能有的错误和容易发生错误的特殊情况，作为选择测试用例的依据。例如：
    - 输入数据和输出数据为0；
    - 输入表格为空格或输入表格只有一行。
  - 例：测试用于线性表排序的程序，推测需要测试的情况：
    - 输入的线性表为空表或表中只含有一个元素；
    - 输入表中所有元素已按顺序排好；
    - 输入表中所有元素已按逆序排好；
    - 输入表中部分或全部元素相同。

随机测试法

- 概述
  - 随机测试指测试用例数据是从被测程序的所有可能输入值中随机选取的，是一种基本的黑盒测试方法。
  - 数据的随机选取可以采用随机模拟的方法，包括采用伪随机数发生器、硬件随机模拟器等方法产生输入数据。
  - 随机测试方法能够获得大量的测试数据。测试人员只需要规定输入变量的取值区间，提供必要的变换机制，使产生的随机数服从预期的概率分布。
- 随机测试法的局限性
  - 随机测试方法不能事先将测试的输入数据存入文档，在排错时难以重现测试中错误发生的过程，不方便进行回归测试。
    - 补救的办法是将随机产生的测试数据记录备用。

Thank you!