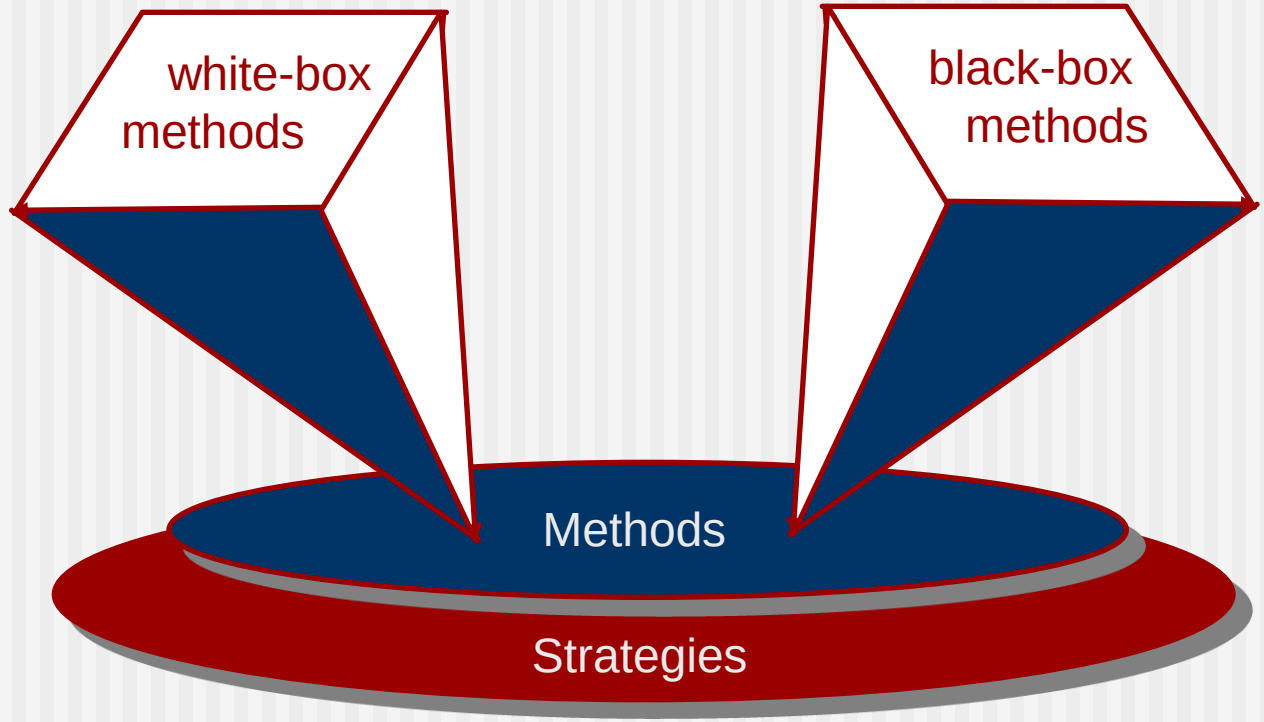
Chapter 10

- Testing Conventional Applications

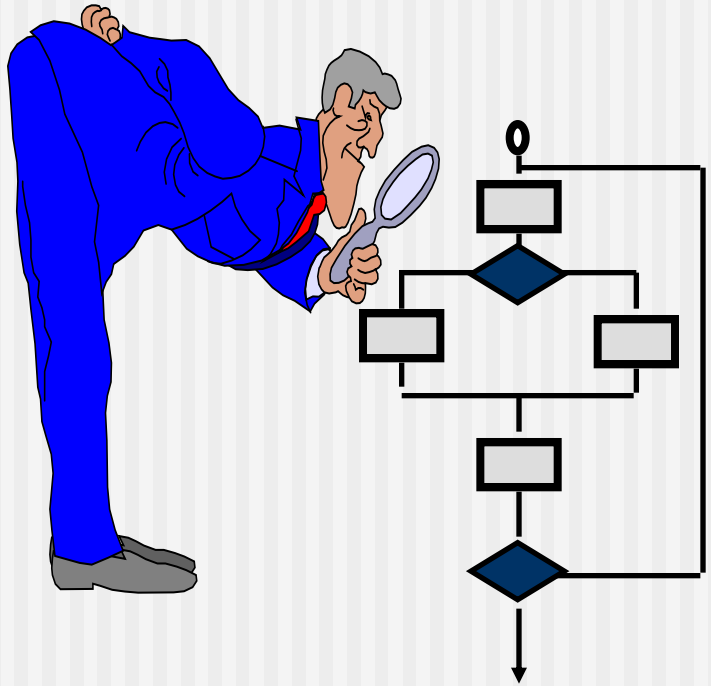
Internal and External Views

- Any engineered product (and most other things) can be tested in one of two ways:
  - Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function;
  - Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh," that is, internal operations are performed according to specifications and all internal components have been adequately exercised.

Software Testing



White-Box Testing



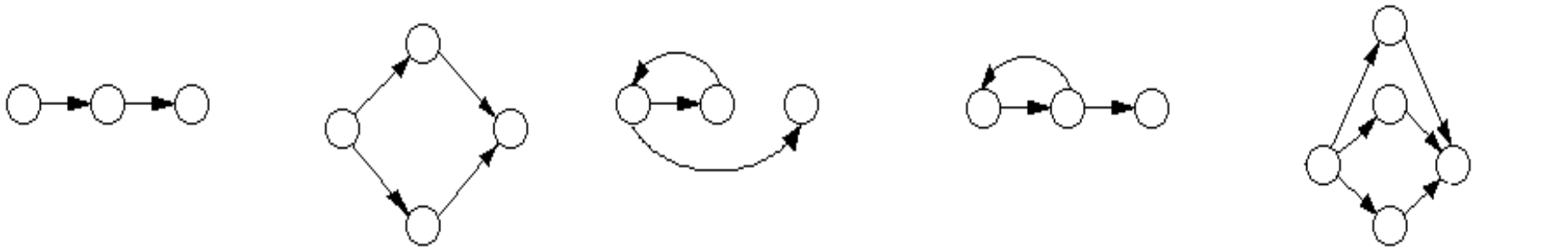
... our goal is to ensure that all statements and conditions have been executed at least once ...

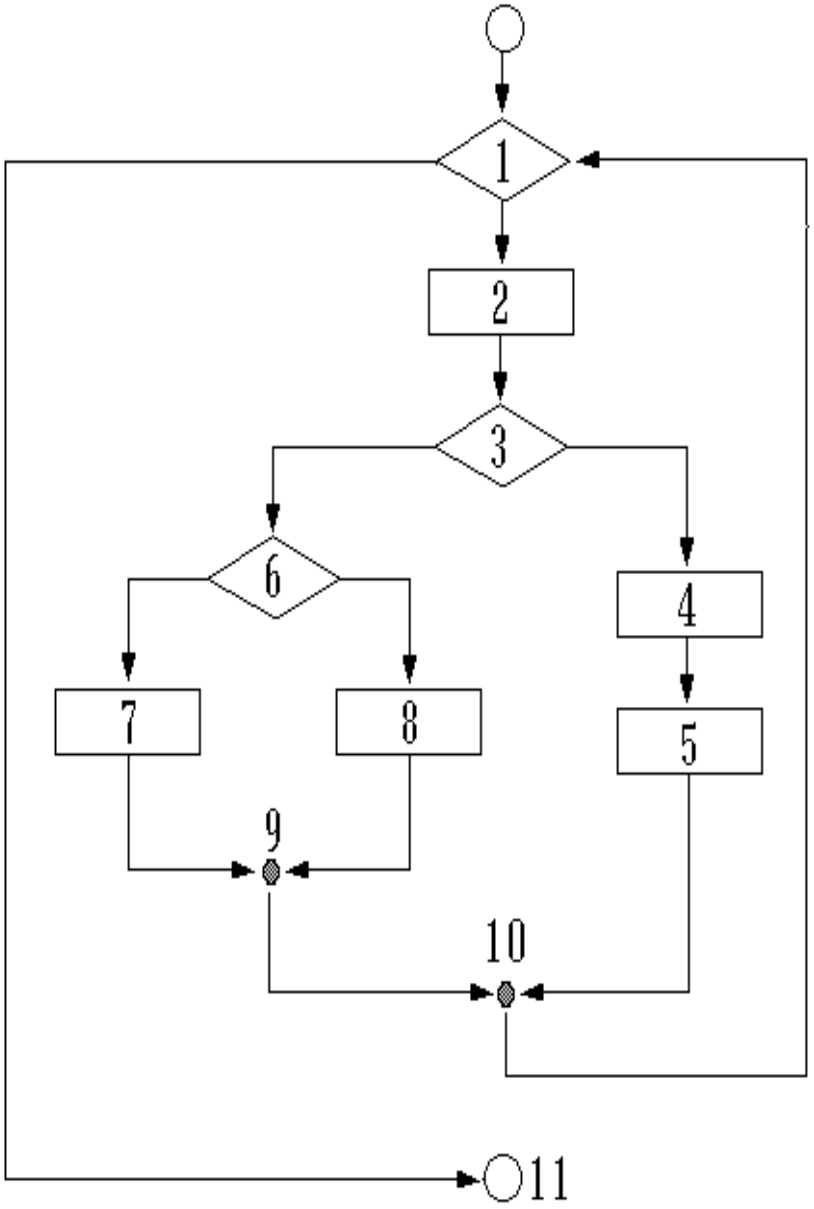
Why Cover?

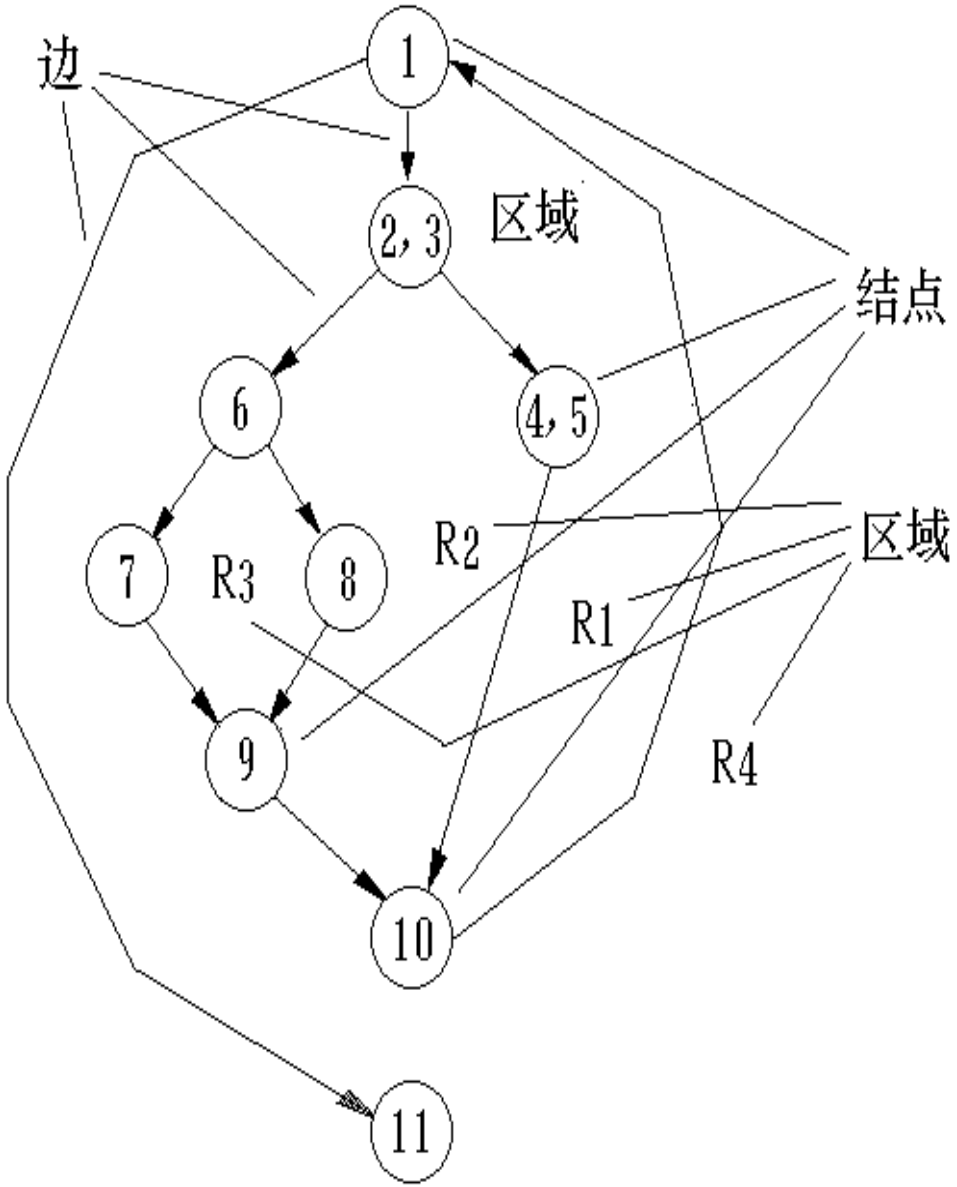
- **logic errors and incorrect assumptions are inversely proportional to a path's execution probability**
- **we often believe that a path is not likely to be executed; in fact, reality is often counter intuitive**
- **typographical errors are random; it's likely that untested paths will contain some**

程序的控制流图

- 符号〇为控制流图的一个结点，表示一个或多个无分支的**PDL**语句或源程序语句。箭头为边，表示控制流的方向。




1
2
3
6
4
7
8
5
9
10
11


边
1
2, 3
区域
结点
6
4, 5
7
R3
8
R2
区域
R1
9
R4
10
11

Basis Path Testing

- 例如，在图示的控制流图中，一组独立的路径是
  path1：1 - 11
  path2：1 - 2 - 3 - 4 - 5 - 10 - 1 - 11
  path3：1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11
  path4：1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11
- 路径 path1，path2，path3，path4组成了控制流图的一个基本路径集。

Basis Path Testing

- 环复杂性以图论为基础，可以通过以下三种方法之一来计算：

1.域的数量与环复杂性相对应。

2.对流图G，环复杂性V(G)定义如下：

V(G)=E-N+2

其中E为流图的边数，N为流图的结点数。

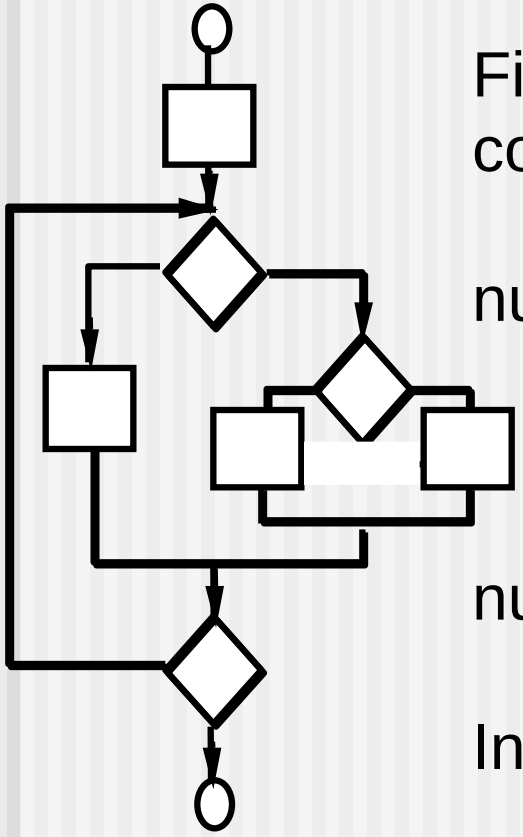
3.对流图G，环复杂性V(G)也可以定义如下：

V(G)=P+1

其中P为包含在流图G中的判定结点数。

V(G)的值提供了组成基本集的独立路径的上界，并由此得出覆盖所有程序语句所需测试数量的上界。

Basis Path Testing



First, we compute the cyclomatic complexity:

number of simple decisions + 1

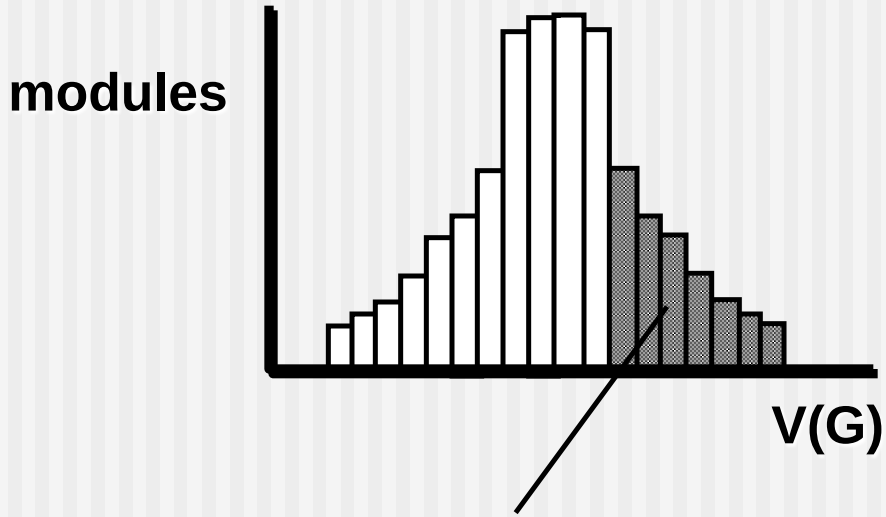
or

number of enclosed areas + 1

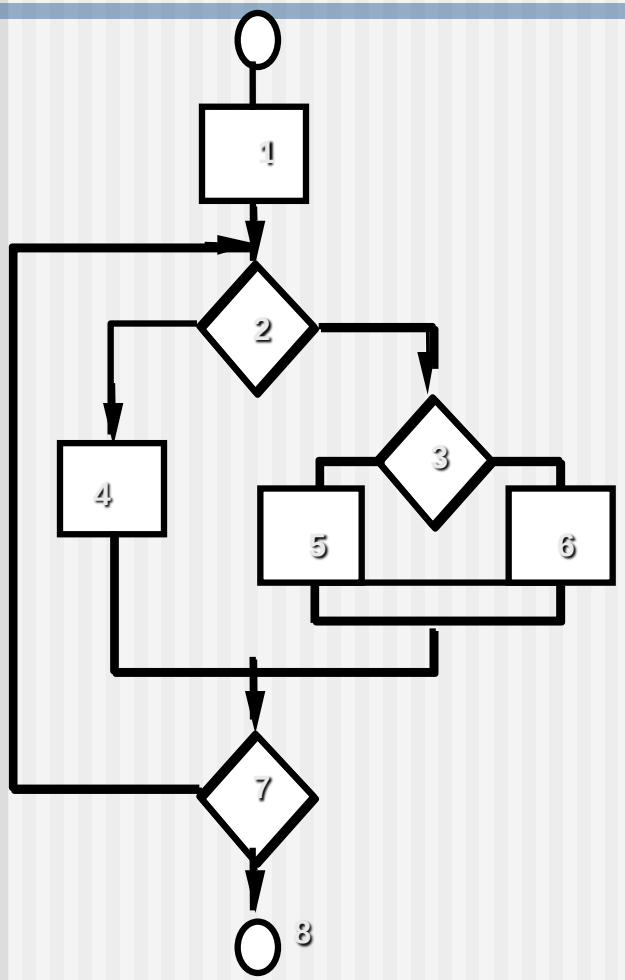
In this case, V(G) = 4

Cyclomatic Complexity

A number of industry studies have indicated that the higher V(G), the higher the probability or errors.



Basis Path Testing



Next, we derive the independent paths:

Since V(G) = 4,

there are four paths

Path 1: 1,2,3,6,7,8

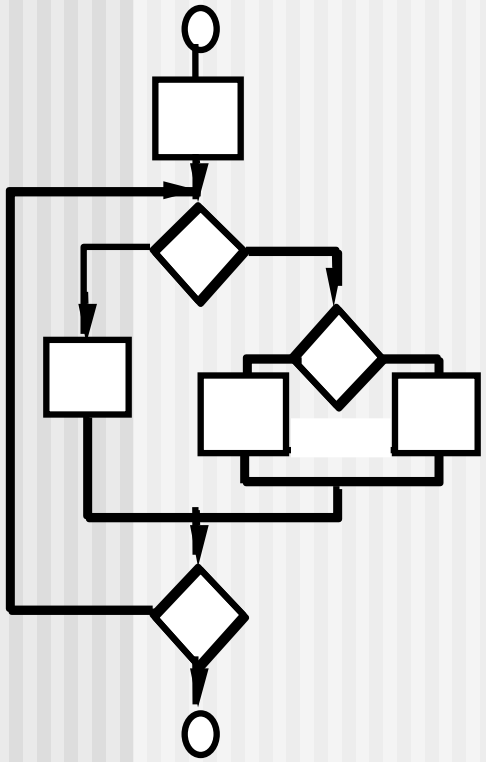
Path 2: 1,2,3,5,7,8

Path 3: 1,2,4,7,8

Path 4: 1,2,4,7,2,4,...7,8

Finally, we derive test cases to exercise these paths.

Basis Path Testing Notes



- you don't need a flow chart, but the picture will help when you trace program paths
- count each simple logical test, compound tests count as 2 or more
- basis path testing should be applied to critical modules

Deriving Test Cases

- 基本路径测试方法可以应用于过程设计或源代码。以下以图15-4中用PDL描述的过程average为例，说明测试用例设计方法中的各个步骤。

Deriving Test Cases

```
PROCEDURE average;

    *  This procedure computes the average of 100 or fewer
       numbers that lie between bounding values; it also computes the
       sum and the total number valid.

    INTERFACE RETURNS average, total.input, total.valid;
    INTERFACE ACCEPTS value, minimum, maximum;

    TYPE value[1:100] IS SCALAR ARRAY;
    TYPE average, total.input, total.valid;
       minimum, maximum, sum IS SCALAR;
    TYPE i IS INTEGER;
1   i = 1;
    total.input = total.valid = 0;
    sum = 0;
    DO WHILE value[i] <> -999 AND total.input < 100
4      increment total.input by 1;
       IF value[i] > = minimum AND value[i] < = maximum
5         THEN increment total.valid by 1;
7                 sum = s sum + value[i]
          ELSE skip
8      ENDIF
       increment i by 1;
9   ENDDO
    IF total.valid > 0
11     THEN average = sum / total.valid;
12     ELSE average = -999;
13  ENDIF
END average
```

图　已标识结点的PDL

Deriving Test Cases

- Using the design or code as a foundation, draw a corresponding flow graph.
- Determine the cyclomatic complexity of the resultant flow graph.
- Determine a basis set of linearly independent paths.
- Prepare test cases that will force execution of each path in the basis set.

❖某些独立路径不能单独进行测试。即遍历路径所需的数据组合不能形成程序的正常流。在这种情况下，这些路径作为另一个路径的一部分进行测试。

Deriving Test Cases

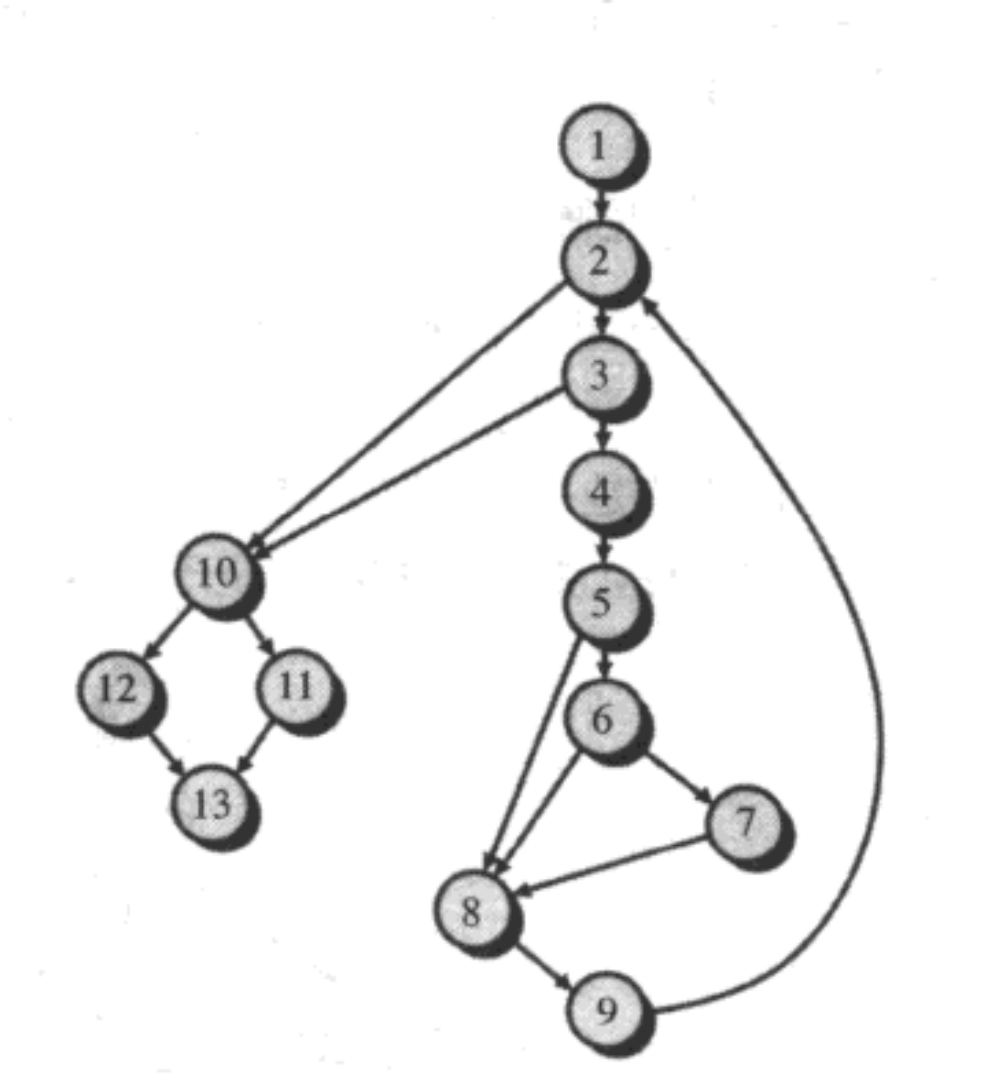
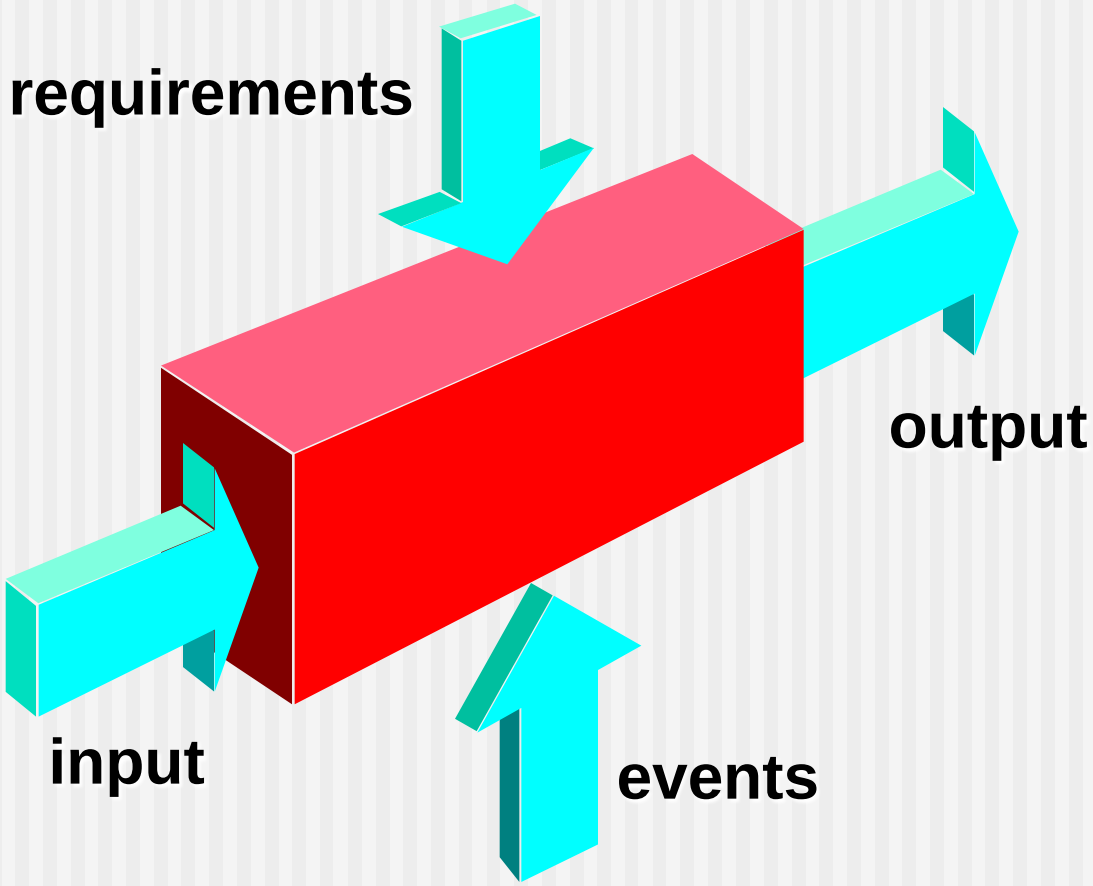


图　过程average的流图

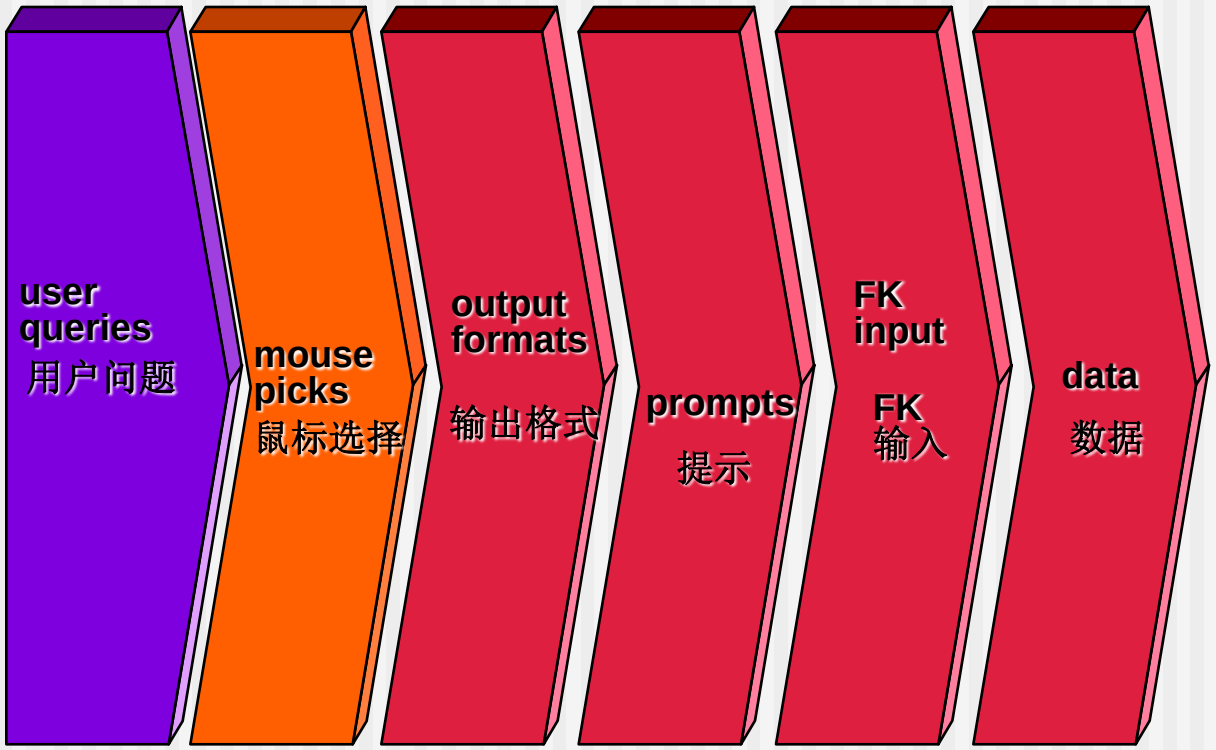
Black-Box Testing



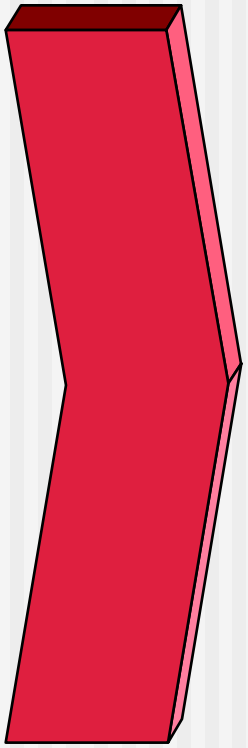
Black-Box Testing

- How is functional validity tested?
- How is system behavior and performance tested?
- What classes of input will make good test cases?
- Is the system particularly sensitive to certain input values?
- How are the boundaries of a data class isolated?
- What data rates and data volume can the system tolerate?
- What effect will specific combinations of data have on system operation?

Equivalence Partitioning



等价类样本



有效数据

用户提供命令

响应系统提示符
文件名

计算数据

物理参数
边界值
初始值

输出数据格式
响应错误消息
图解数据（例如，鼠标选择）

无效数据

程序边界范围外的数据
物理上不可能的数据
在错误的地点提供适当的值

Equivalence Partitioning

- 划分
  - 指互不相交的一组子集，这些子集的并是整个集合。
  - 对测试的意义：*完备性、无冗余性*。

$A_1, A_2, \ldots, A_n$是集合A的子集

$A_1, A_2, \ldots, A_n$是集合A的一个划分

$A_1 \cup A_2 \cup \ldots \cup An = A$ 且

$A_i \cap A_j \ldots = \Phi \ (i \mathrel{!}= j)$

Equivalence Partitioning

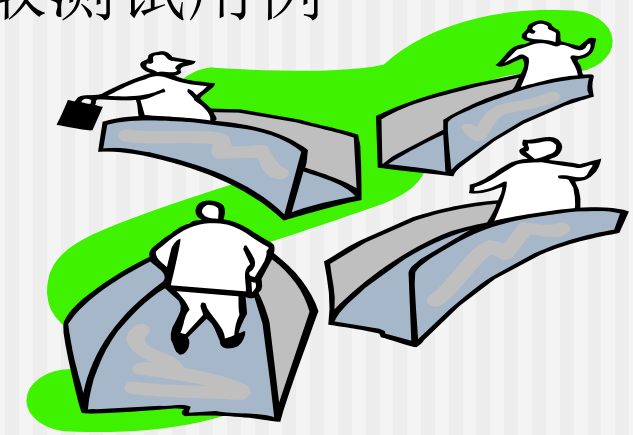
- **等价类**
  - *等价类是指某个输入域的子集合*。在该子集合中，各个输入数据对于*揭示程序中的错误都是等效的*。
- **等价类划分**
  - 等价类划分是一种典型的黑盒测试方法。
  - 这一方法完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例

Equivalence Partitioning

- 等价类合理地假设：某个等价类的代表值，与该等价类的其他值，对于测试来说是等价的。因此，可以把全部的输入数据划分成若干的等价类，在每一个等价类中取一个数据来进行测试。
- 优点：*能以较少的具有代表性的数据进行测试，而取得较好的测试效果*。

Equivalence Partitioning

- 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）和选取测试用例两步。



Equivalence Partitioning

- 等价类的划分有两种不同的情况：
  - ① **有效等价类：**是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。
  - ② **无效等价类：**是指对于程序的规格说明来说，是不合理的，无意义的输入数据构成的集合。
- 在设计测试用例时，要同时考虑*有效等价类和无效等价类的设计*

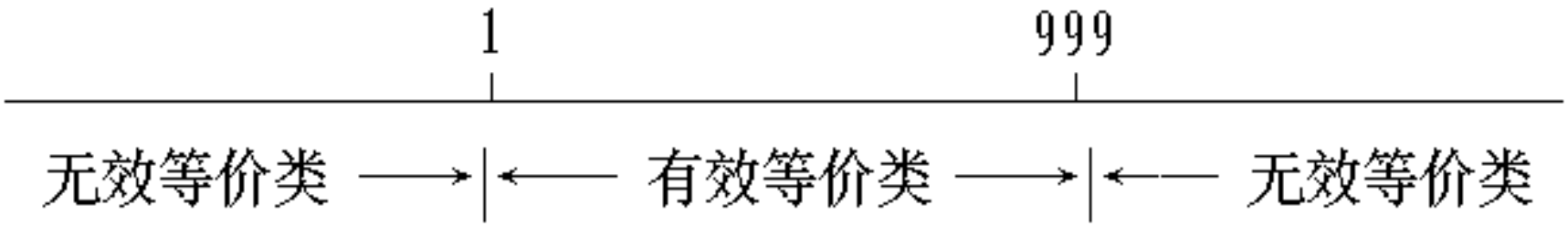
Equivalence Partitioning

- 例如，在程序的规格说明中，对输入条件有一句话：

  “…… 项数可以从1到999 ……”

  则有效等价类是“1≤项数≤999”

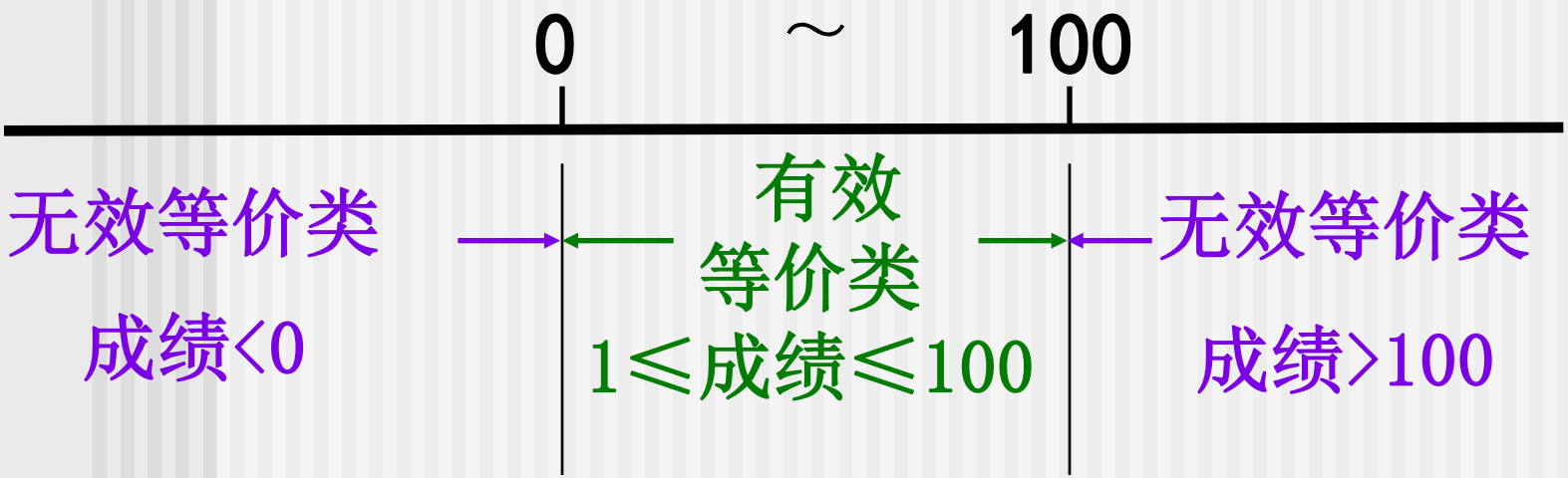
  两个无效等价类是“项数＜1”或“项数＞999”。在数轴上表示成：



划分等价类的规则

- **(1)**如果输入条件规定了取值范围，可定义一个有效等价类和两个无效等价类。

例: 输入值是学生成绩，范围是0～100



划分等价类的规则

- **(2)**如果输入条件代表集合的某个元素，则可定义一个有效等价类和一个无效等价类。
  - 如：某程序涉及到标识符，其输人条件规定“标识符应以字母开头…”，则“以字母开头者”作为有效等价类，“以非字母开头”为无效等价类。
- **(3)**如果输入条件是一个布尔量，则可以确立一个有效等价类和一个无效等价类；

划分等价类的规则

- **(4)**如规定了输入数据的一组值，且程序对不同输入值作不同处理，则每个允许的输入值是一个有效等价类，并有一个无效等价类**(**所有不允许的输入值的集合**)**。
  - 例：输入条件说明学历可为**:**专科、本科、硕士、博士四种之一，则分别取这四个值作为四个有效等价类，另外把四种学历之外的任何学历作为无效等价类。

划分等价类的规则

- **(5)**如果规定了输入数据必须遵循的规则，可确定一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则**)**。

- **(6)**如已划分的等价类各元素在程序中的处理方式不同，则应将此等价类进一步划分成更小的等价类。

用等价类划分法设计测试用例步骤：

- **(1)**形成等价类表，每一等价类规定一个唯一的编号；
- **(2)**设计一测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类，重复这一步骤，直到所有有效等价类均被测试用例所覆盖；
- **(3)**设计一新测试用例，使其只覆盖一个无效等价类，重复这一步骤直到所有无效等价类均被覆盖；

Equivalence Partitioning

- 例:

  某报表处理系统要求用户输入处理报表的日期，日期限制在2001年1月至2005年12月，即系统只能对该段期间内的报表进行处理，如日期不在此范围内，则显示输入错误信息。

  系统日期规定由年、月的6位数字字符组成，前四位代表年，后两位代表月。

  如何用等价类划分法设计测试用例, 来测试程序的日期检查功能？

第一步：等价类划分

“报表日期”输入条件的等价类表

输入等价类	有效等价类	无效等价类
报表日期的类型及长度	6位数字字符(1)	有非数字字符 (4) 少于6个数字字符 (5) 多于6个数字字符 (6)
年份范围	在2001～2005之间(2)	小于2001 (7) 大于2005 (8)
月份范围	在1～12之间(3)	小于1 (9) 大于12 (10)

第二步：为有效等价类设计测试用例

对表中编号为1,2,3的3个有效等价类用一个测试用例覆盖：

测试数据	期望结果	覆盖范围
200105	输入有效	等价类(1)(2)(3)

第三步：为每一个无效等价类设计至少一个测试用例

测试数据	期望结果	覆盖范围
001MAY	输入无效	等价类(4)
20015	输入无效	等价类(5)
2001005	输入无效	等价类(6)
200005	输入无效	等价类(7)
200805	输入无效	等价类(8)
200100	输入无效	等价类(9)
200113	输入无效	等价类(10)

不能出现相同的测试用例

本例的10个等价类至少需要8个测试用例

Questions

- 1、What is white-box testing ? What are the methods of white-box testing?
- 2、What is black-box testing ? What are the methods of black-box testing?