Topics in Software Dynamic White-box Testing
Part 1: Control-flow Testing

[Reading assignment: Chapter 7, pp. 105-122 plus many things in slides that are not in the book …]

白盒测试分类

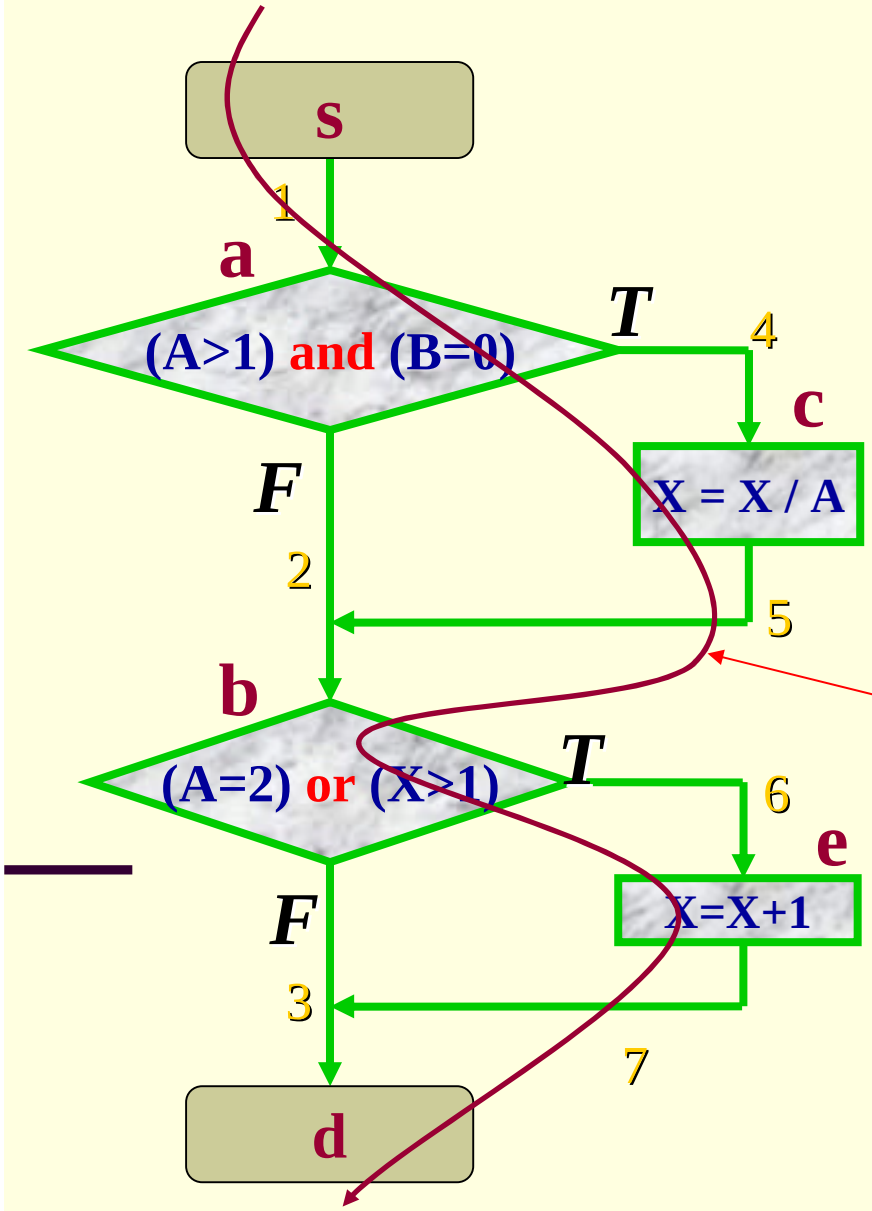
- **ControlFlow-testing**
  - 逻辑分支覆盖法
    - 语句覆盖
    - 判定覆盖
    - 条件覆盖
    - 判定/条件覆盖
    - 条件组合覆盖
  - 路径法
    - 路径覆盖
      - 基本（独立）路径测试法
- **DataFlow-testing**

14.2.1.1语句覆盖

- 语句覆盖就是设计若干个测试用例，运行被测程序，使得每一可执行语句至少执行一次。
- 这种覆盖又称为点覆盖，它使得程序中每个可执行语句都得到执行，但它是最弱的逻辑覆盖，效果有限，必须与其它方法交互使用。

To be continue…

程序框图



源程序

```
PROCEDURE Example(A,B:real; X:real );
Begin
    IF (A>1) AND (B=0) THEN
        X:= X / A;
    IF ( A=2 ) OR (X>1) THEN
        X:=X+1
END;
```

I. A=2, B= 0, X=4 ---- sacbed

14.2.1.2判定覆盖

- 判定覆盖就是设计若干个测试用例，运行被测程序，使得程序中每个判断的取真分支和取假分支至少经历一次。判定覆盖又称为分支覆盖。
- 判定覆盖只比语句覆盖稍强一些，但实际效果表明，只是判定覆盖，还不能保证一定能查出在判断的条件中存在的错误。因此，还需要更强的逻辑覆盖准则去检验判断内部条件。

To be continue…

判定覆盖用例

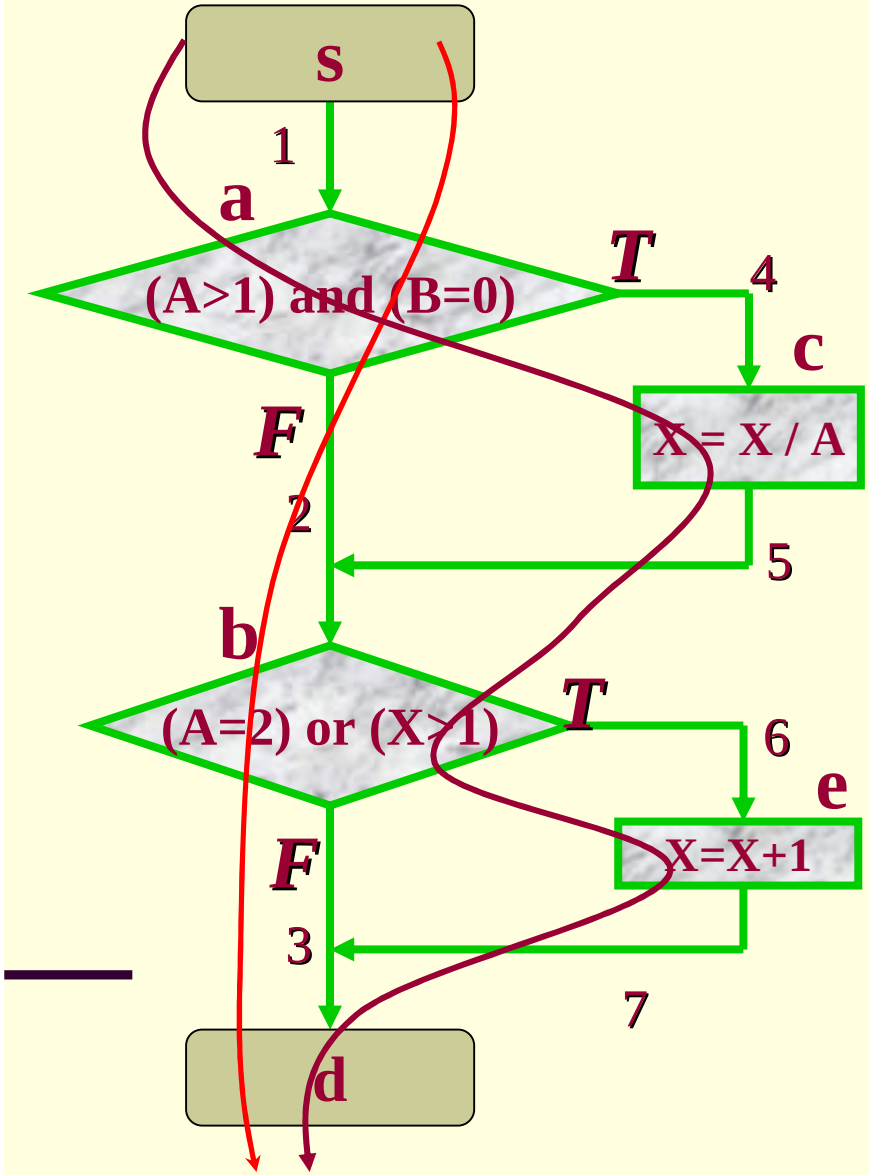
- 若再给出两组测试用例是：
- A =2 B=0 X=3测试用例3
- A =1 B=0 X=1 测试用例

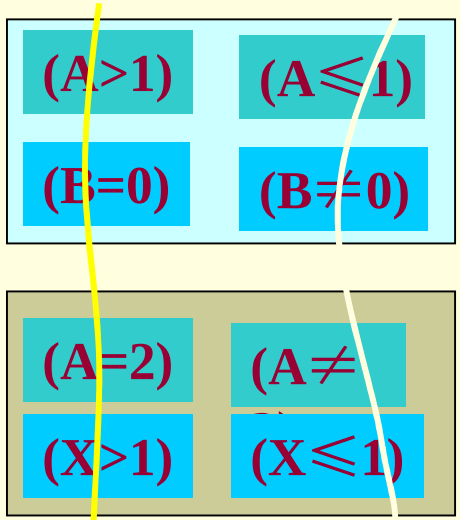
测试用例	A B X	(A>1) AND (B=0)	(A=2) OR (X>1)	执行路径	预期结果X
测试用例3	2 0 3	真(T)	真(T)	sacbed	2.5
测试用例4	1 0 1	假(-T)	假(-T)	sabd	1

14.2.1.3条件覆盖

- 条件覆盖就是设计若干个测试用例，运行被测程序，使得程序中每个判断的每个条件的可能取值至少执行一次。
- 条件覆盖深入到判定中的每个条件，但可能不能满足判定覆盖的要求。

To be continue…





Ⅰ: A=2, B=0,X=4: sacbed

Ⅱ: A=1, B=1,X=1: sabd

测试用例			通过路径	满足的条件
A	B	X		
2	0	4	sacbed	T1,T2,T3,T4
1	1	1	sabd	$\overline{T1},\overline{T2},\overline{T3},\overline{T4}$

条件覆盖

- 条件覆盖通常比判定覆盖强，因为它使判定表达式中每个条件都取到了两个不同的结果，判定覆盖却关心整个判定表达式的值。
- 但也可能有相反的情况：虽然每个条件都取到了不同值，但判定表达式却始终只取一个值。
- 条件覆盖不一定包含判定覆盖
- 判定覆盖也不一定包含条件覆盖。
- 为解决这一矛盾，需要对条件和判定兼顾。

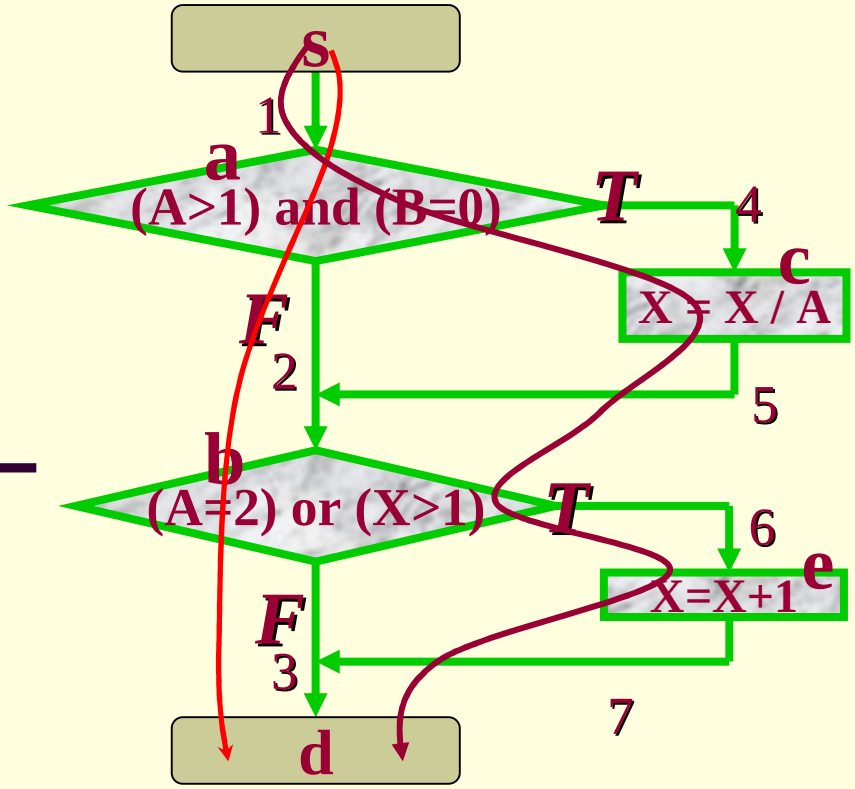
To be continue…

14.2.1.4判定-条件覆盖

- 既然判定覆盖不一定包含条件覆盖，条件覆盖也不一定包含判定覆盖，就自然会提出一种能同时满足两种覆盖标准的逻辑覆盖，这就是判定/条件覆盖。

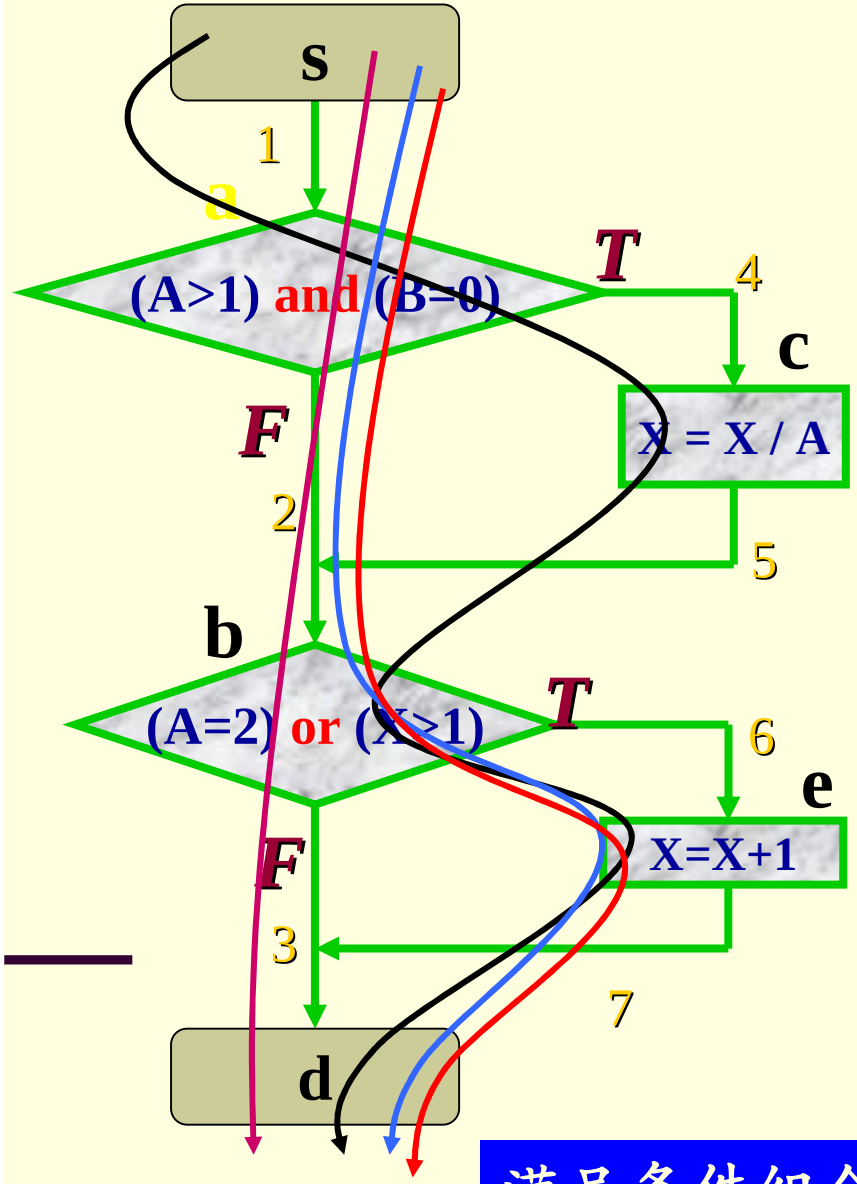
To be continue…

测试用例	A B X	执行路径	覆盖条件	(A>1)AND(B=0)	(A=2)OR(X>1)	预期结果 X
测试用例 1	2 0 3	sacbed	T1,T2,T3,T4	真（T）	真（T）	2.5
测试用例 2	1 1 1	sabd	−T1,−T2, −T3,−T4	假（−T）	假（−T）	1



14.2.1.5条件组合覆盖

- 条件组合覆盖就是设计足够的测试用例，运行被测程序，使得每个判断的所有可能的条件取值组合至少执行一次。
- 显然，满足“条件组合覆盖”的测试用例是一定满足“判定覆盖”、“条件覆盖”和“判定－条件覆盖”的。
- 这是一种相当强的覆盖准则，可以有效地检查各种可能的条件取值的组合是否正确。它不但可覆盖所有条件的可能取值的组合，还可覆盖所有判断的可取分支，但可能有的路径会遗漏掉。测试还不完全。



1. A＞1, B＝0
2. A＞1, B≠0
3. A≯1, B＝0
4. A≯1, B≠0

I. A=2,B=0,X=4

II. A=2,B=1,X=1

5. A＝2, X＞1
6. A＝2, X≯1
7. A≠2, X＞1
8. A≠2, X≯1

III. A=1,B=0,X=2

IV. A=1,B=1,X=1

覆盖路径

I: sacbed
II: sabed
III: sabed
IV: sabd

满足条件组合覆盖标准的测试数据，也一定满足判定覆盖、条件覆盖和判定/条件覆盖标准。

白盒测试技术

逻辑分支覆盖法

- 语句覆盖
- 判定覆盖
- 条件覆盖
- 判定/条件覆盖
- 条件组合覆盖

路径法

- 路径覆盖
- 基本（独立）路径测试法

- 路径法
  - 路径覆盖
  - 基本（独立）路径测试法

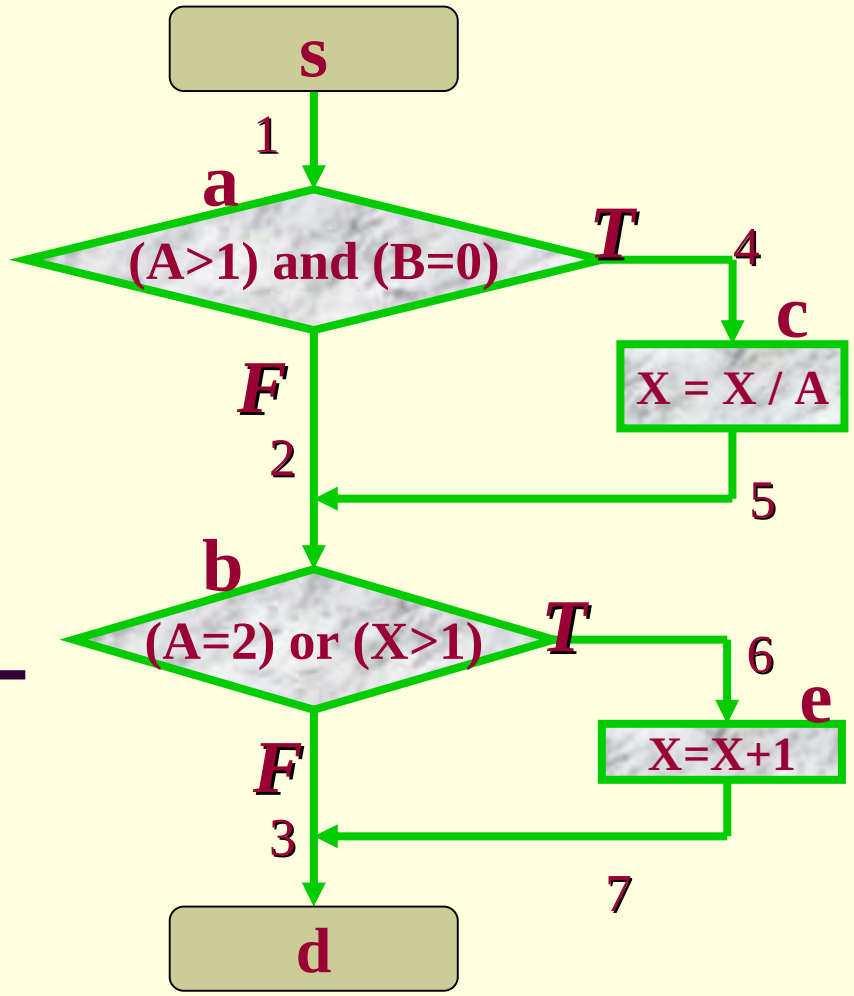
14.2.1.6路径覆盖

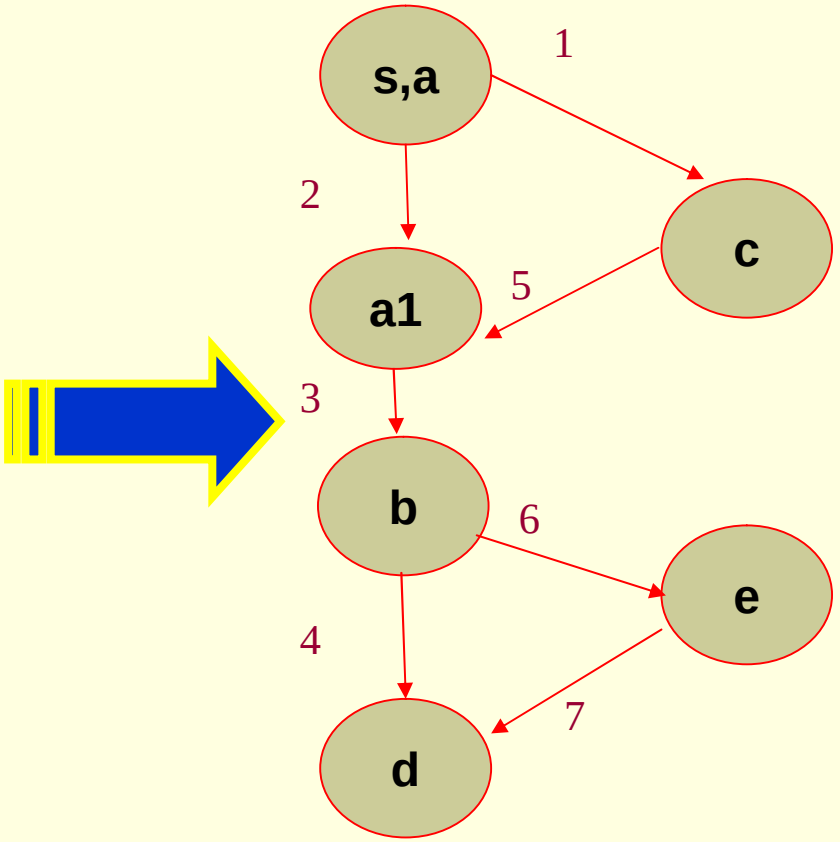
- 路径测试就是设计足够的测试用例，覆盖程序中所有可能的路径。这是最强的覆盖准则。但在路径数目很大时，真正做到完全覆盖是很困难的，必须把覆盖路径数目压缩到一定限度。

几个约定

- 一个判定构成流图的一个节点
- 顺序结构的几个结点可合并为一个结点
- 对应于每个判定分支结构，在结束时增加一个虚拟的汇聚结点，多个汇聚结点可合并
- 区域包括封闭区域和开放区域

流图


S
1
a
(A>1) and (B=0)
T
4
c
X = X / A
F
2
5
b
(A=2) or (X>1)
T
6
e
X=X+1
F
3
7
d


s,a
1
2
c
a1
5
3
b
6
e
4
7
d

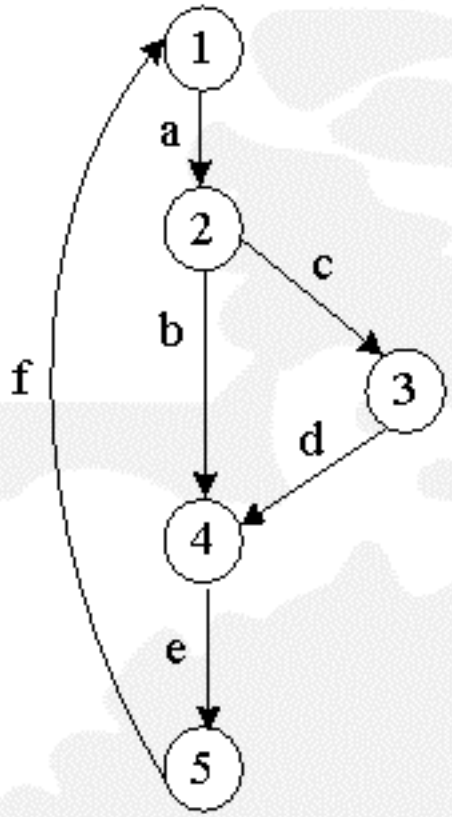
路径表示方法

- 用边表示

  abefacde

- 用结点表示

1-2-4-5-1-2-3-4-5



路径表达式概括表达路径

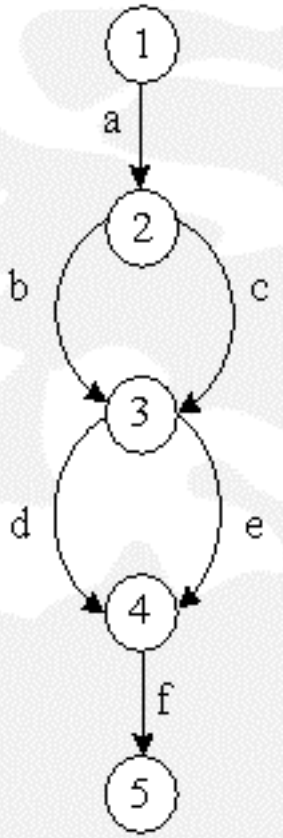
- 概括表示所有路径

  a(b+c)(d+e)f

- 拆分

  abdf+abef+acdf+acef

  所有的四条路径



路径覆盖

路径覆盖是相当强的逻辑覆盖，它保证程序中每条可能的路径都至少执行一次，因此更具代表性，暴露错误的能力也比较强。但为了做到路径覆盖，只需考虑每个判定式的取值组合，并没有检验表达式中条件的各种可能组合。如果将路径覆盖和条件组合覆盖结合起来，可以设计出检错能力更强的测试数据。

独立路径

- 定义
  - 从入口到出口的路径，至少经历一个从未走过的边。这样形成的路径叫独立路径。
- 优点
  - 减少路径数量
  - 包含所有的边和结点
- 缺点
  - 简化循环结构

基本路径测试步骤

① 根据程序的逻辑结构画出程序框图
② 根据程序框图导出流图
③ 计算流图G的环路复杂度V(G)
④ 确定只包含独立路径的基本路径集
⑤ 设计测试用例

程序框图 ==> 流图 ==> 基本路径 ==> 测试用例

白盒路径测试技术

环复杂度计算法：

三种方法之一：

1. 流图的区域数量应该对应于环复杂度

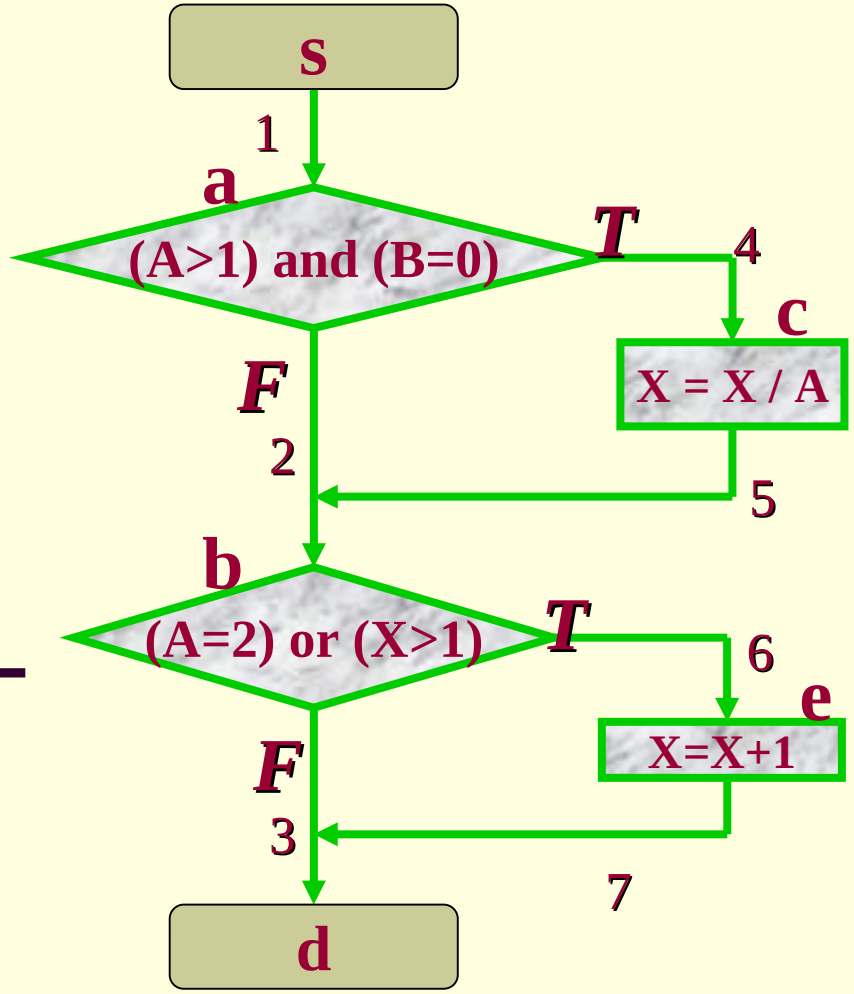
2. 给定流图G的环复杂度V(G)定义为：V(G)=E-N+2

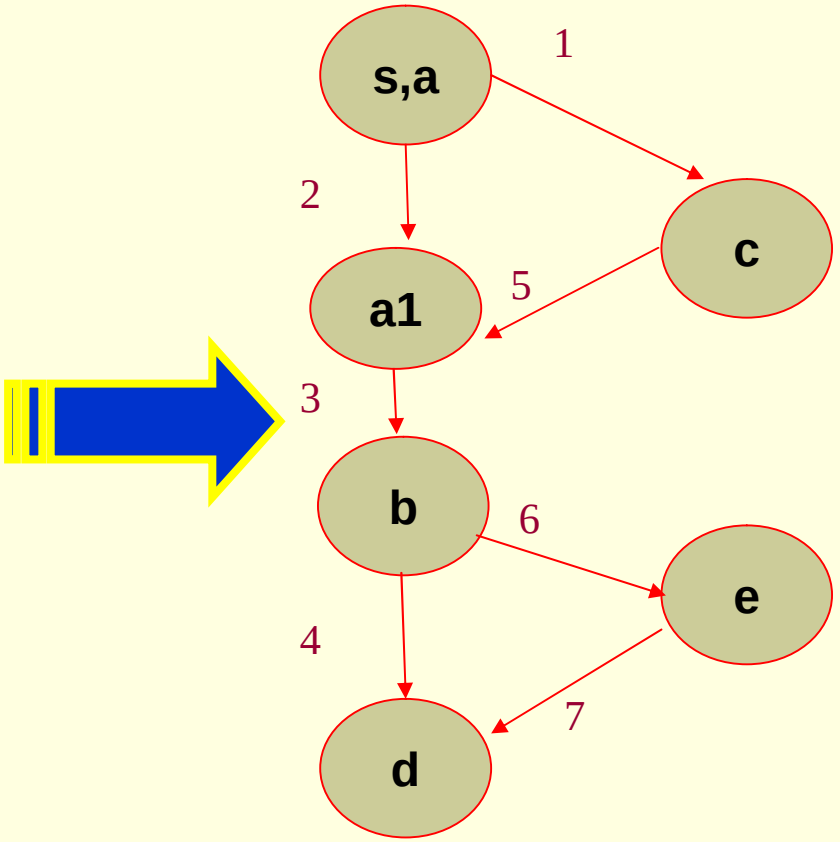
   其中：E为流图中的边数量，N为流图中的节点数量

3. 给定流图G的环复杂度V(G)也可以定义为：V(G)=P+1

   其中：P为流图中的判断节点数量

独立路径


S
1
a
(A>1) and (B=0)
T
4
c
X = X / A
F
2
5
b
(A=2) or (X>1)
T
6
e
X=X+1
F
3
7
d


s,a
1
2
c
a1
5
3
b
6
e
4
7
d

独立路径＝3

步骤3： 确定基本路径的集合

流图的环形复杂度正好＝基本路径的数目

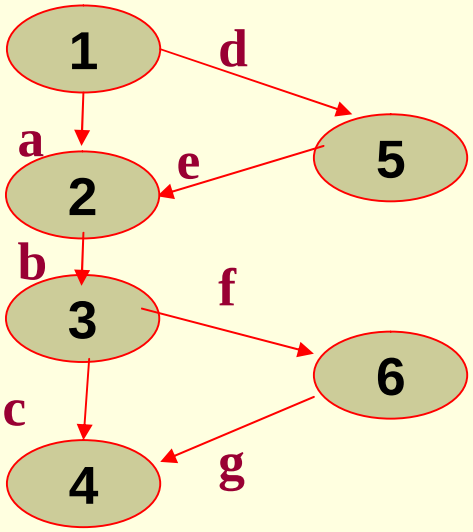
确定只包含独立路径的基本路径集

Path1: 节点表示1-2-3-4(边表示abc)

Path2: 节点表示1-5-2-3-4(边表示debc)

Path3: 节点表示1-2-3-6-4(边表示abfg)

一条新路径必须包含一条新边



步骤4：对每条基本路径设计测试用例

测试用例			通过路径	预期结果
A	B	X		
1	1	1	**1-2-3-4**	X=1
3	0	1	**1-5-2-3-4**	X=0.33
1	0	3	**1-2-3-6-4**	X=4