CHAPTER 7

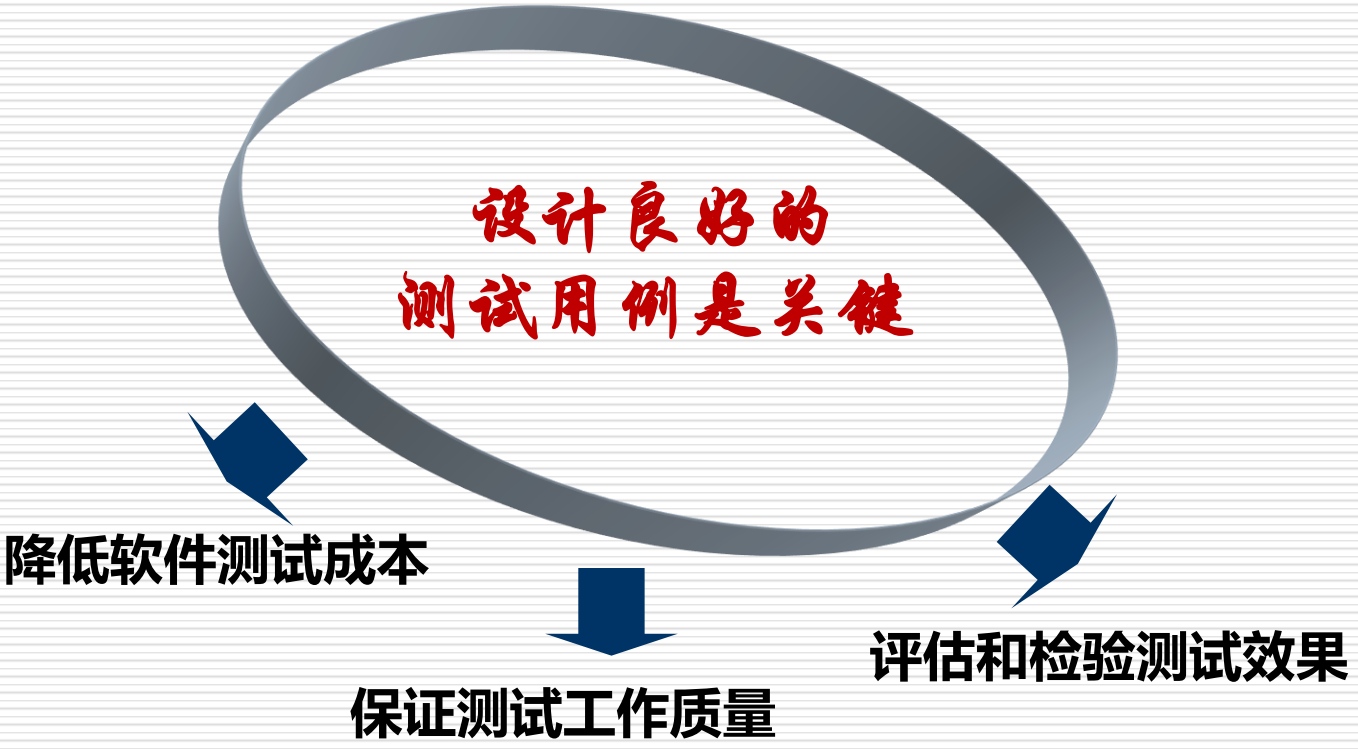
Software Testing

测试方法

- 白盒测试
- 黑盒测试

测试用例：

（一组输入、输出参数）



测试用例的重要性

指导人们系统地进行测试

- ✓ 临时性发挥也许会有灵感出现，但是多数情况下会感觉思维混乱，甚至一些功能根本没有测到，而另一些功能已经重复测过几遍。
- ✓ 测试用例可以帮助你理清头绪，进行比较系统的测试，不会有太多的重复，也不会让你的测试工作产生遗漏。

有效发现缺陷，提高测试效率

- ✓ 测试不可能是完备的而且受到时间约束，测试用例可以帮助你分清先后主次，从而更有效地组织测试工作。
- ✓ 编写测试用例之后需要标识重要程度和优先级，以便在时间紧迫的情况下有重点地开展测试工作。

测试用例的重要性

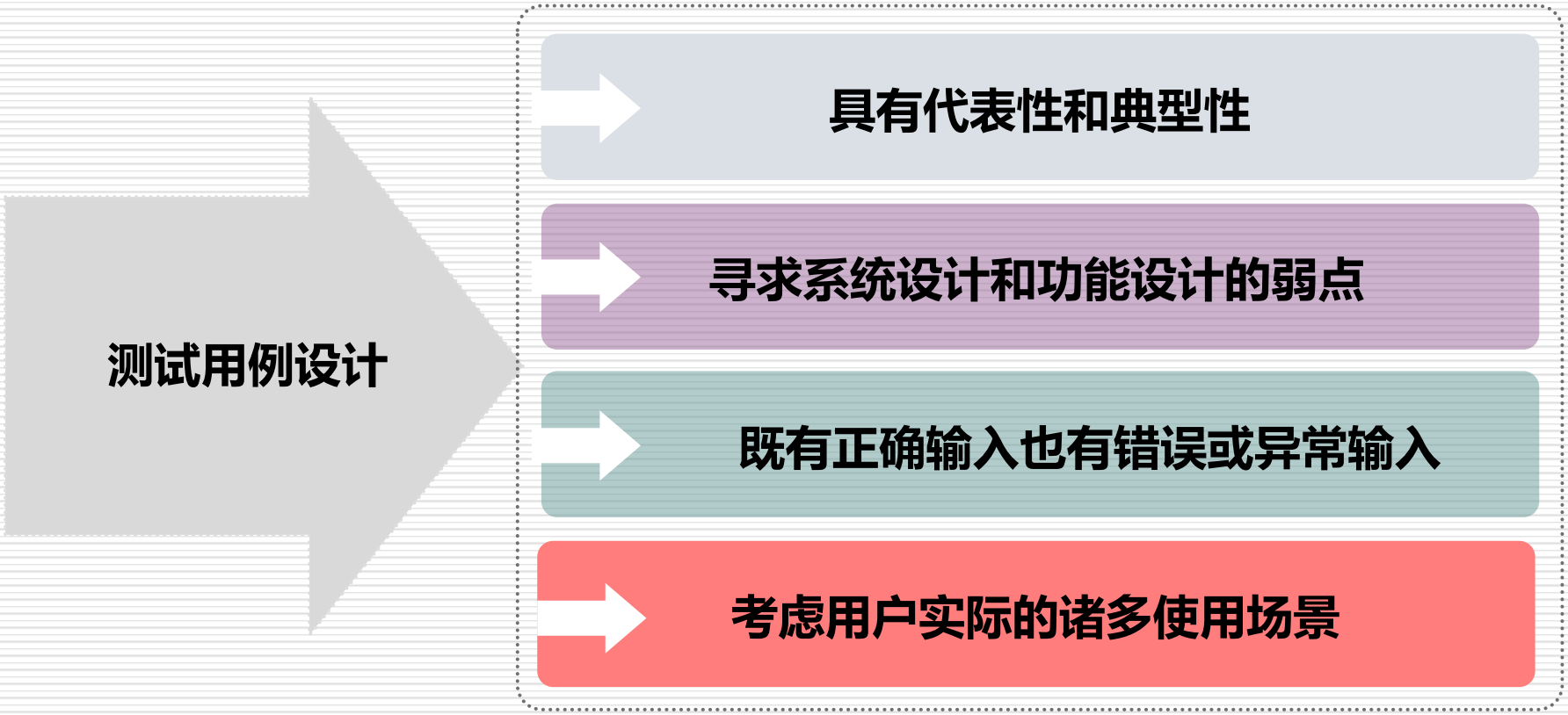
作为评估和检验的度量标准

- ✓ 测试用例的通过率和软件缺陷的数量是检验软件质量的量化标准，通过对测试用例的分析和改进可以逐步完善软件质量，不断提高测试的水平。
- ✓ 测试用例也可以用于衡量测试人员的工作量、进度和效率，从而更有效地管理和规划测试工作。

积累和传递测试的经验与知识

- ✓ 测试用例不是简单地描述一种具体实现，而是描述处理具体问题的思路。设计和维护测试用例有助于人们不断积累经验和知识，通过复用测试用例可以做到任何人实现无品质差异的测试。

测试用例设计要求



- 一项有挑战性的工作

案例讨论：纸杯测试

人们在日常生活中经常使用一次性纸杯，请根据自己的生活常识，提出尽可能多的测试用例，并进一步给出设计建议。

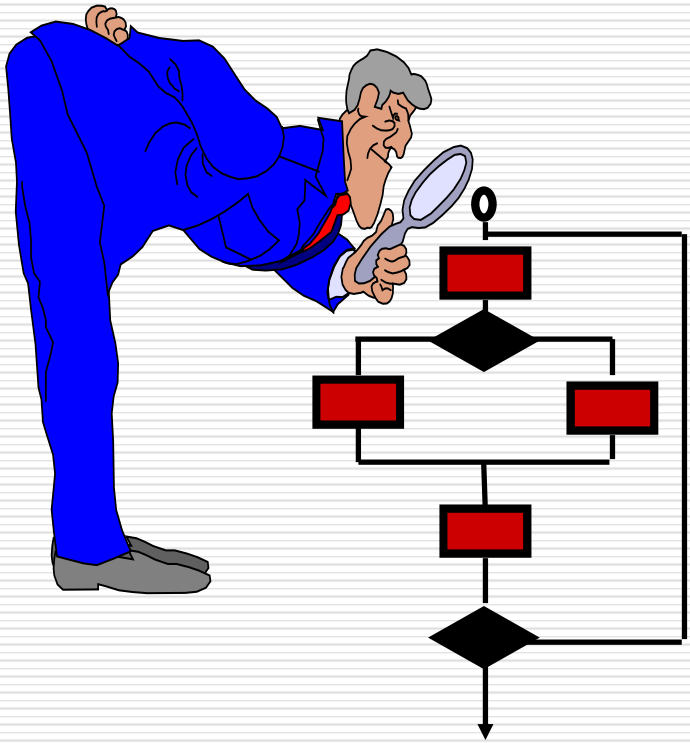
怎么测，测哪些方面？

案例讨论：纸杯测试

- 漏不漏；是否太软，不能捏；时间久了水渗透？
- 盛水，还能盛其它饮料？盛冰水、盛温水、盛开水？
- 纸杯边缘刺人么？能不能放稳？颜料会不会染到嘴上？
- 纸杯的大小，手好不好拿，能否拿得下？微风吹会不会倒？
- 纸杯上图案外观是否有问题（出口）？
- 小孩可能咬纸杯，会咬碎，会吃下去么？
- 纸杯有标记，可区别，避免混淆？
- ……

White-Box Testing

- **Goal:**
  - **Ensure that all statements and conditions have been executed at least once**
- **Derive test cases that:**
  - **Exercise all independent execution paths**
  - **Exercise all logical decisions on both true and false sides**
  - **Execute all loops at their boundaries and within operational bounds**
  - **Exercise internal data structures to ensure validity**



Why White-Box Testing?

- More errors in ‘special case’ code which is infrequently executed
- Control flow can’t be predicted accurately in black-box testing
- Type errors can happen anywhere!

白盒测试

又称：

- 内部测试 internal test
- 开盒测试 open-box test
- 结构测试 structure test
- 玻璃盒测试 glass-box test
- 基于覆盖的测试 coverage based

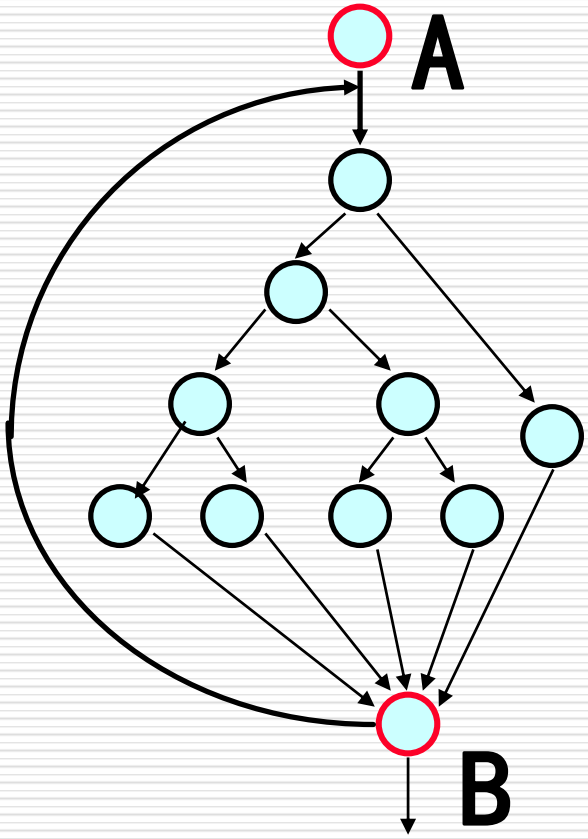
根据被测程序的逻辑结构设计测试用例，力求提高测试覆盖率

Exhaustive white-box testing (infeasible)

例子：设程序含4个分支，循环次数≤20，从A到B的可能路径

$5^1+5^2+\ldots+5^{19}+5^{20}$

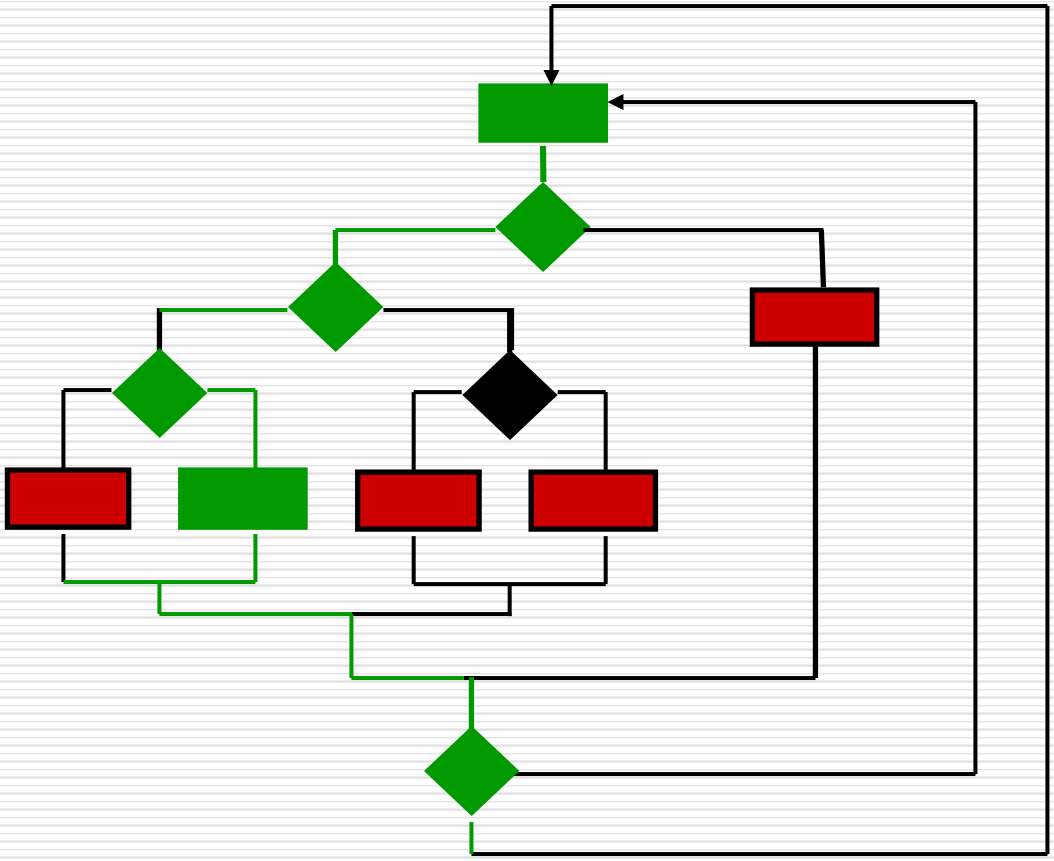
$\approx 10^{14}$



执行时间：设测试一次需 2 ms

穷举测试需 5 亿年

Selective Testing (feasible)



Test a carefully selected execution path. Cannot be comprehensive

Test design

- ✓ **Selection means design.**
- ✓ **It is necessary that software testing need to design. Good testing depend on good test design**
- ✓ **A good test case is one that can find an as-yet-undiscovered error .**
- ✓ **How to design test cases?**

Test case design in white-box testing

Rules:

(1) 语句覆盖

(2) 判定覆盖

(3) 条件覆盖

(4) 判定/条件覆盖

(5) 条件组合覆盖

(6) 路径覆盖

(7) 点覆盖

(8) 边覆盖

Statement Coverage

□定义：

Designing a series of test cases and running them so that every statement is executed at least once.

An example：

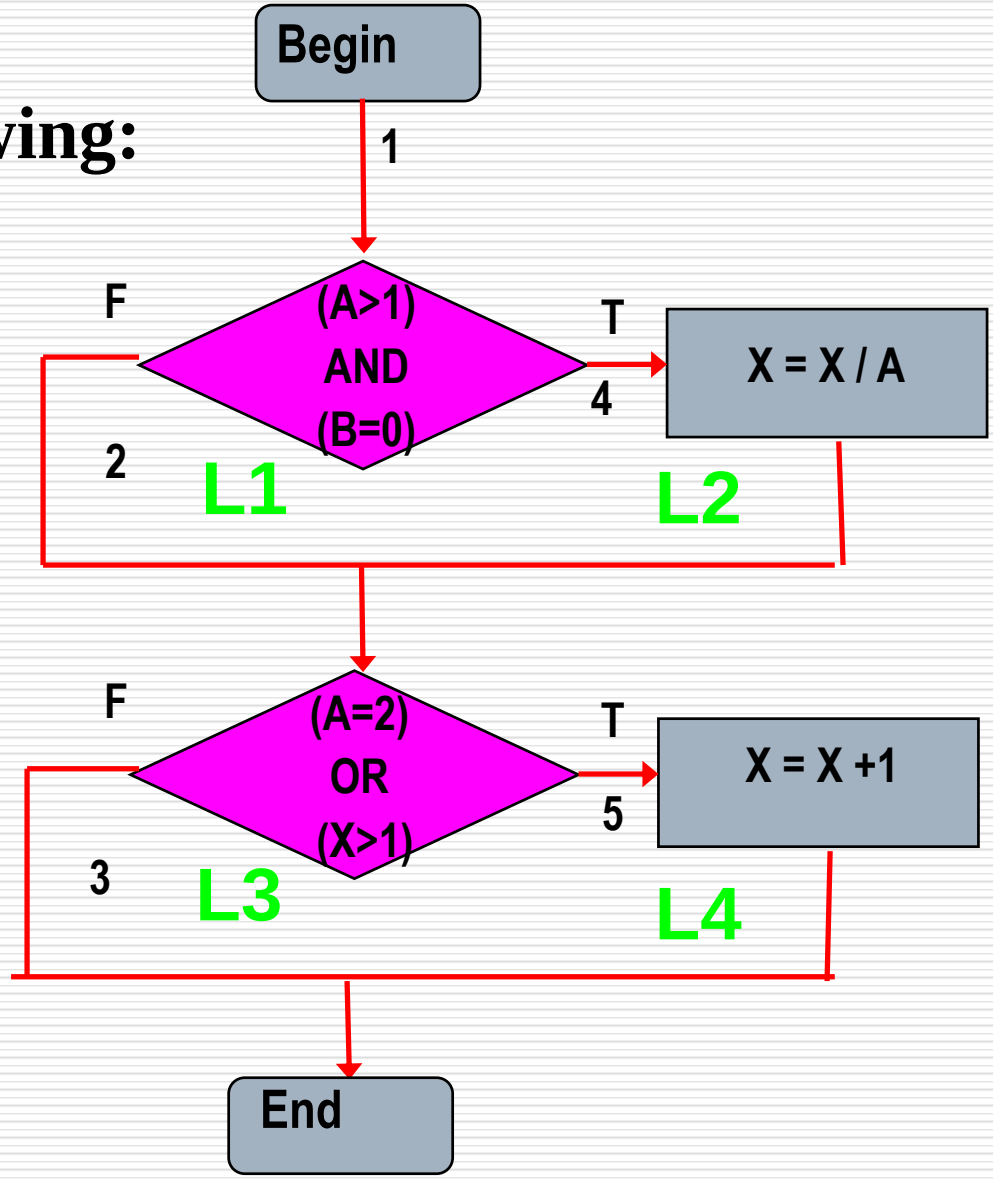
```
PROCEDURE EXAMPLE(A, B: REAL, VAR  X, REAL);
    Begin
L1:    IF ( A>1) AND ( B=0 )    THEN
L2:              X:=X / A;
L3:    IF ( A=2 ) OR ( X > 1)     THEN
L4:              X:= X +1
     End;
```

Design test data as following:

Variables: A, B, X

A=2, B=0, X=4

语句覆盖分析：



语句覆盖的特点

A=1, B=1, X=1 ✗ 不符合要求

A=4, B=0, X=8 不唯一

Disadvantage：

- **No guarantee that all outcomes of branches are properly tested.**
- **Weakest coverage**

Decision Coverage (Branch Coverage)

- 判断覆盖定义：

Designing a series of test cases and running them so that every branch (decision) is executed at least once.

举例： 前面的例子

Decisions are diamond box

All branches are 1,2,3,4,5

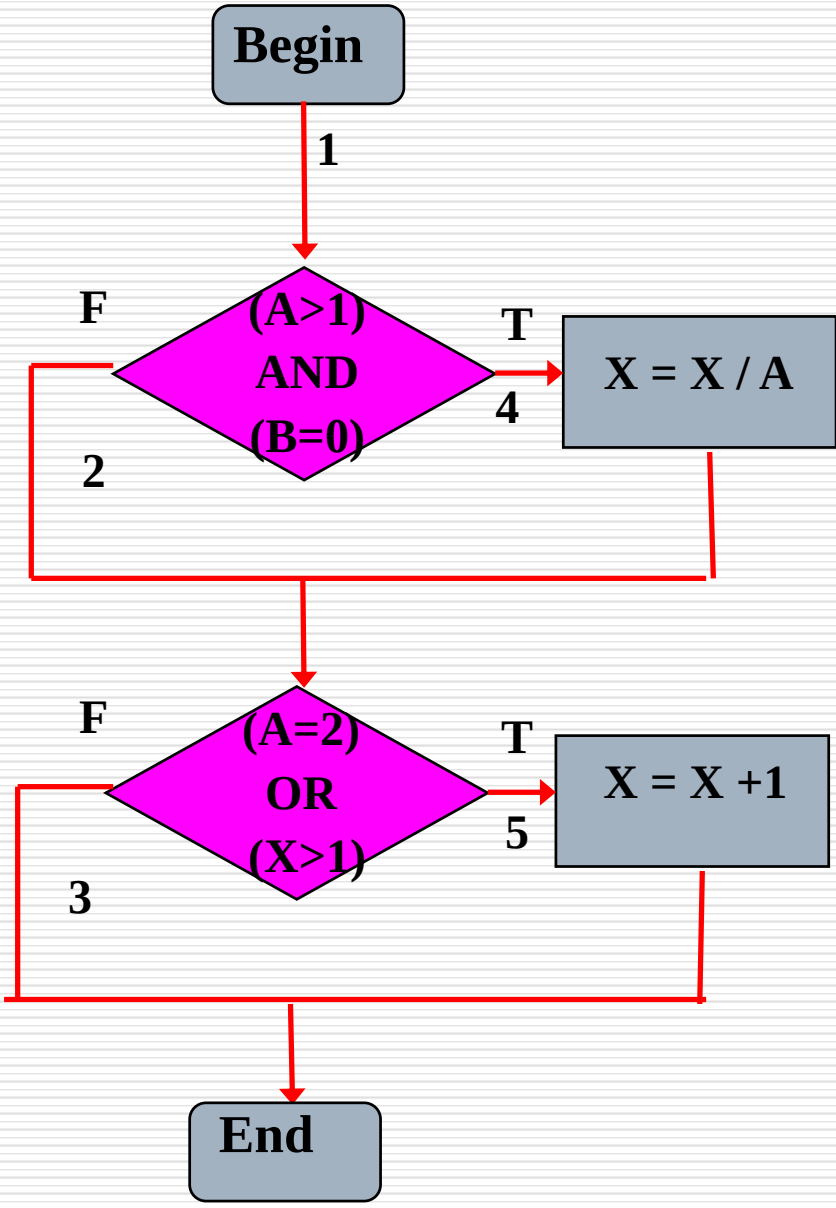
设计：

(1) (A,B,X) = (3, 0, 3)

covering 1, 4, 3 (T,F)

(2) (A,B,X) = (2, 1, 1)

covering 1, 2, 5 (F,T)



Weakness：

- Stronger than statement coverage.
- No guarantee that all outcomes of conditions are properly tested.

Condition Coverage

- 条件覆盖定义：

Designing a series of test cases and running them so that every condition in each decision is executed at least once.

举例： 前面的例子

two decisions:

(A>1) AND (B=0)

(A=2) OR (X>1)

all conditions:（每个逻辑表达式）

A>1, A<=1, B=0, B≠0;

A=2, A≠2, X>1, X<=1;

designing test cases:

A=2, B=0, X=4, covering A>1, B=0, A=2, X>1

A=1, B=1, X=1, covering A<=1, B≠0,A≠2, X=1

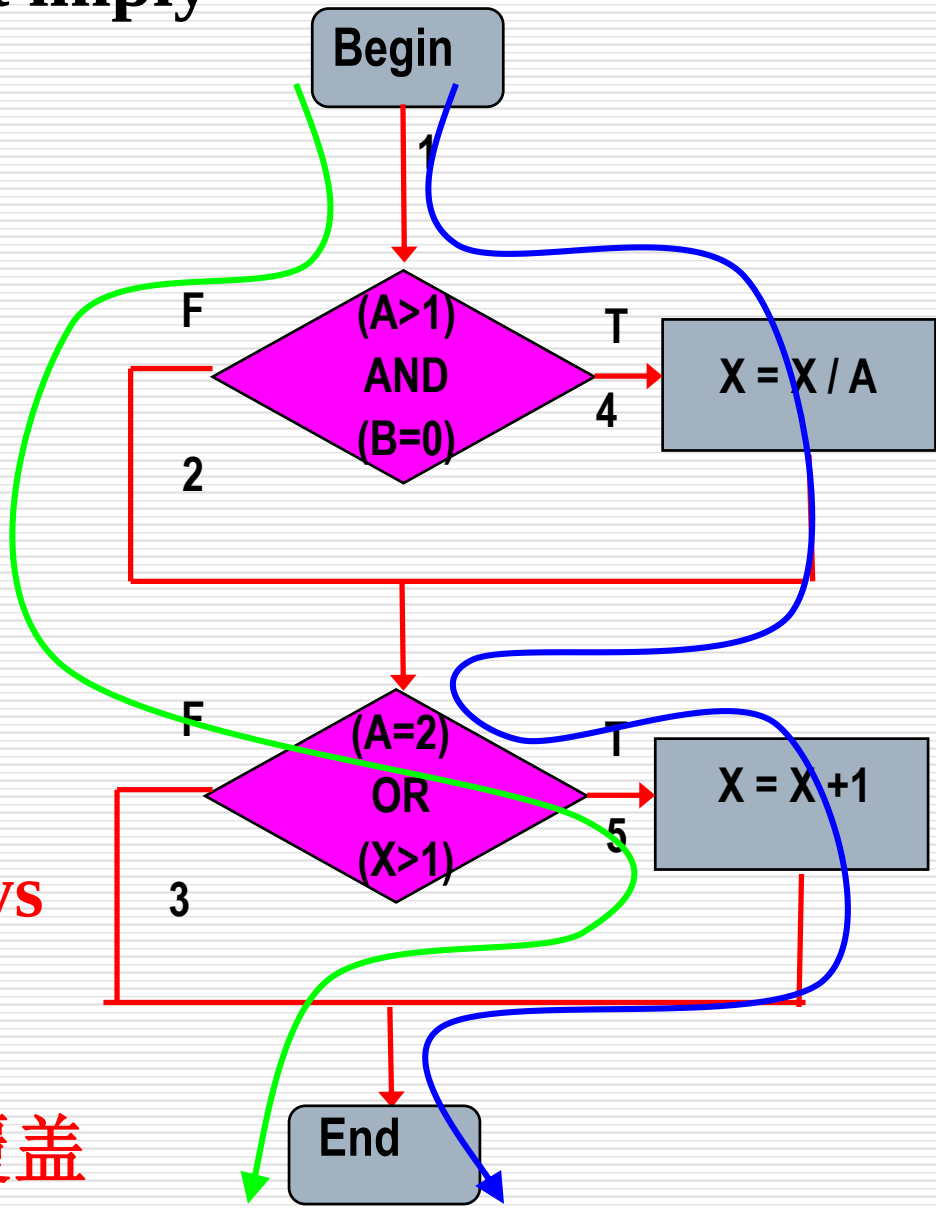
discussion:

✓ **Condition coverage does not imply decision coverage.**

A=2, B=0, X=1 covering
A>1, B=0, A=2, X<=1, path 1-4-5;

A=1, B=1, X=2 covering
A<=1, B≠0, A≠2, X>1, path 1-2-5

The second decision is always true. 3 分支没被覆盖。



✓ 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

Decision & Condition Coverage

- 定义：

Satisfying decision coverage and condition coverage at the same time

Designing a series of test cases and running them so that every branch is executed at least once, and every condition in each decision is executed at least once.

An example of test cases

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

A=1, B=1, X=1,

? Covered

All 2 decisions

4 branches

All 4 condition expressions

8 instances

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

Covered:

1, 4, 5

A>1, B=0, A=2, X>1

A=1, B=1, X=1,

Covered:

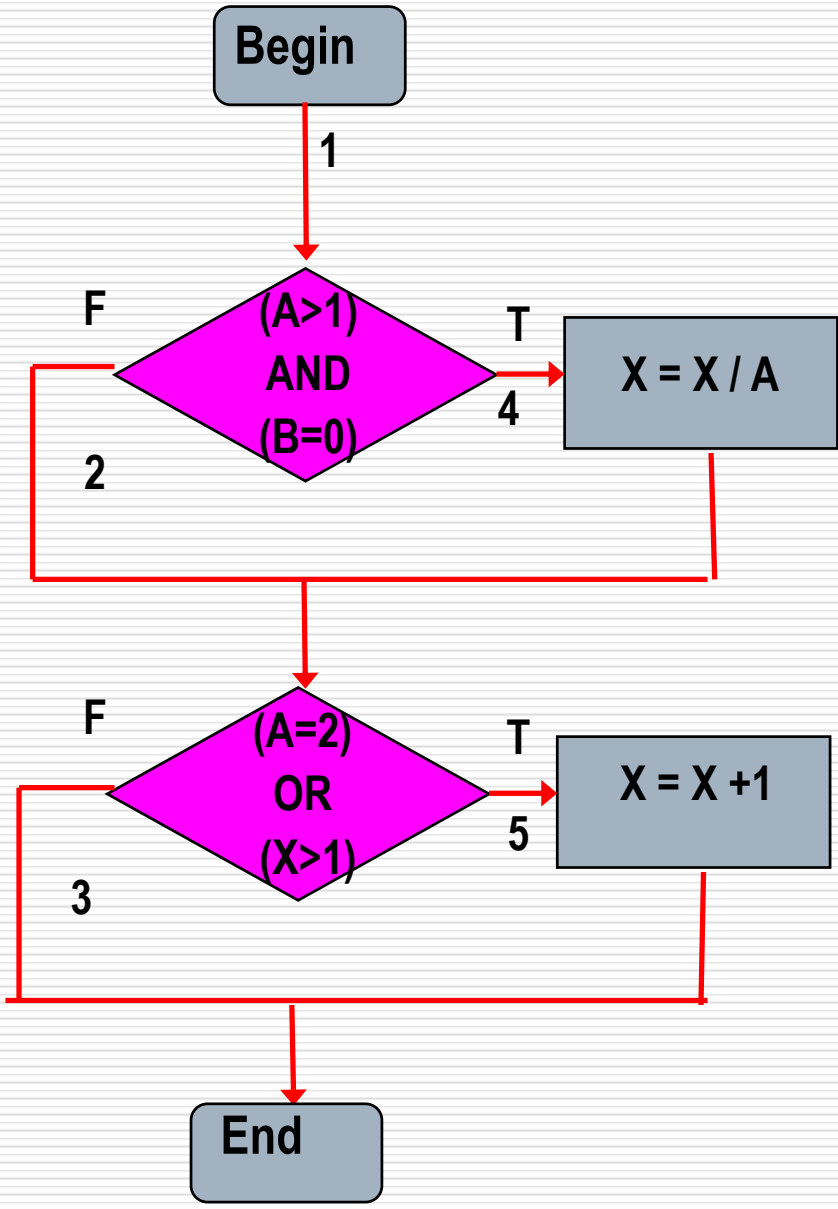
1, 2, 3

A<=1, B≠0, A≠2, X<=1

Any question ?

A>1, B ≠0

A=2, X<=1



Condition Combination Coverage

- 条件组合覆盖定义：

Designing test cases as many as possible and running them so that all condition combination in each decision are executed.

All condition combinations:

(1) A>1, B=0

(2) A>1, B≠0

(3) A<=1, B=0

(4) A<=1, B≠0

(5) A=2, X>1

(6) A=2, X<=1

(7) A≠2, X>1

(8) A≠2, X<=1

条件组合覆盖

所有可能的条件取值组合至少执行一次

A>1, B=0

A>1, B≠0

A≯1, B=0

A≯1, B≠0

A=2, X>1

A=2, X≯1

A≠2, X>1

A≠2, X≯1

✓All condition combinations:

(1) A>1, B=0

(2) A>1, B≠0

(3) A<=1, B=0

(4) A<=1, B≠0

(5) A=2, X>1

(6) A=2, X<=1

(7) A≠2, X>1

(8) A≠2, X<=1

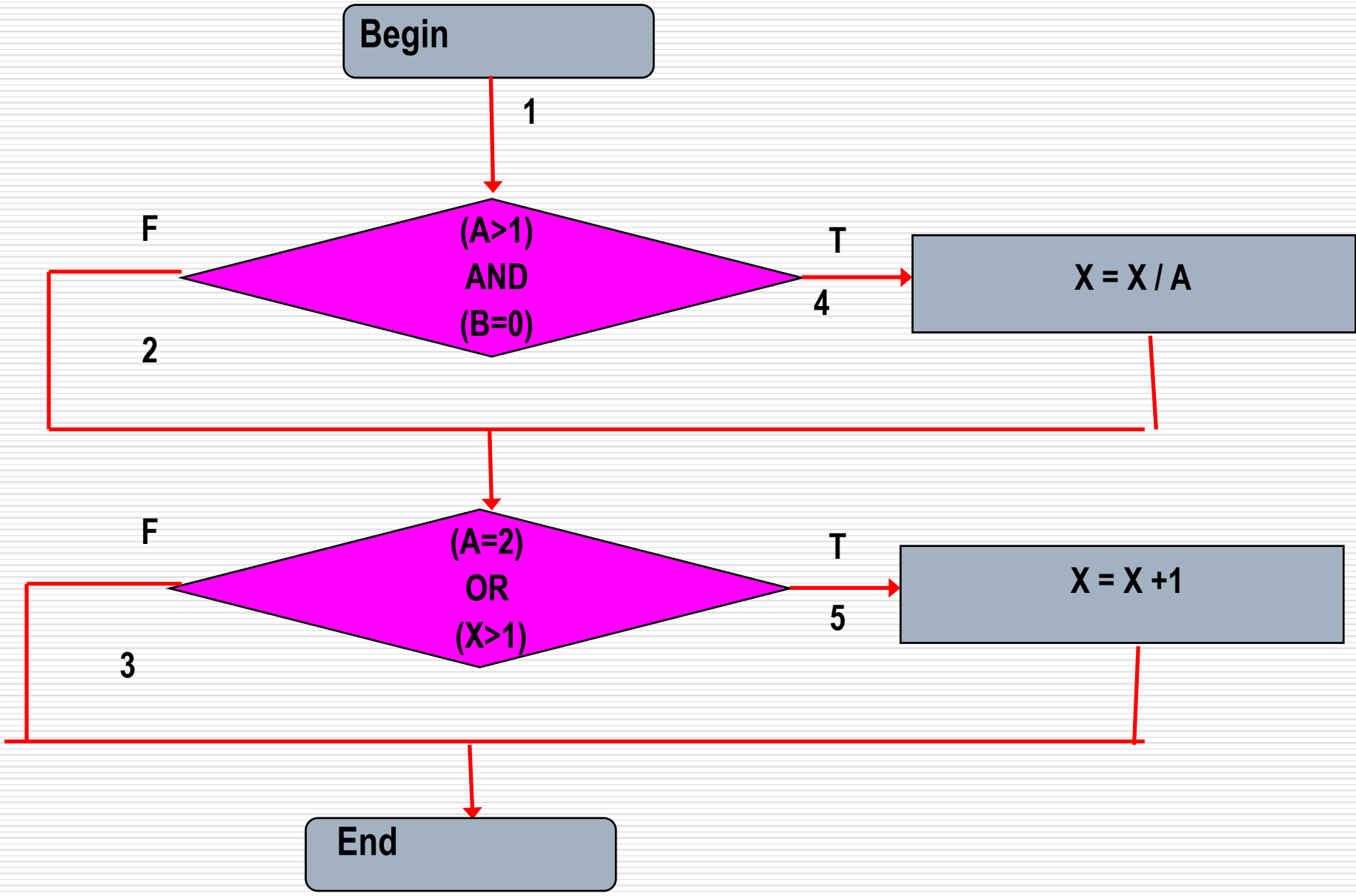
✓test cases:

A=2, B=0, X=4, covered (1) (5), path 1-4-5

A=2, B=1, X=1, covered (2) (6), path 1-2-5

A=1, B=0, X=2, covered (3) (7), path 1-2-5

A=1, B=1, X=1, covering (4) (8), path 1-2-3


Begin
1
F
(A>1)
AND
(B=0)
T
4
X = X / A
2
F
(A=2)
OR
(X>1)
T
5
X = X +1
3
End

Discussion

Condition combination coverage

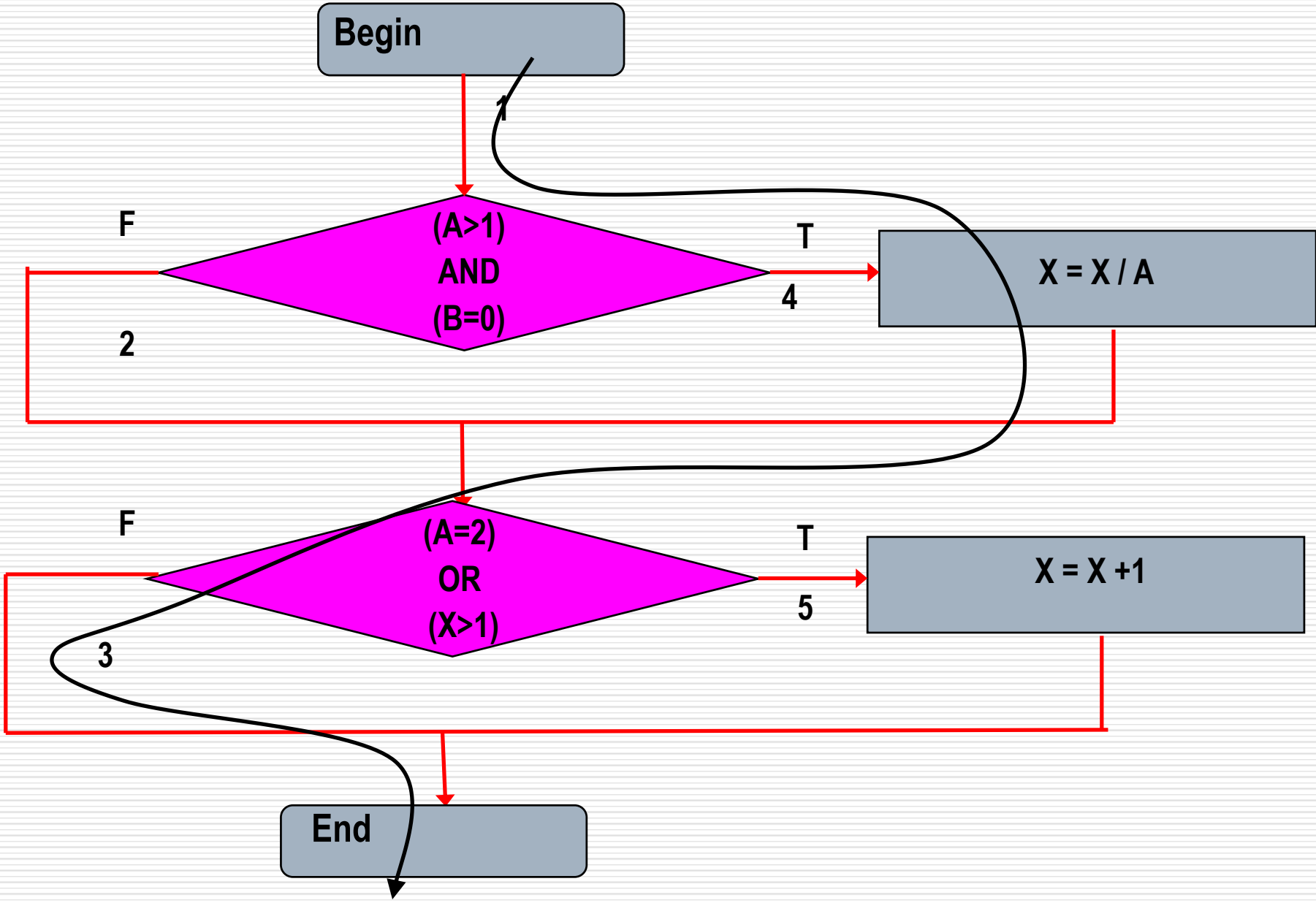
→ **decision coverage**

→ **condition coverage**

→ **decision & condition coverage**

But can condition combination coverage ensure that all path are covered ?

Path 1-4-3 in the example before


Begin
1
F
(A>1)
AND
(B=0)
T
X = X / A
4
2
F
(A=2)
OR
(X>1)
T
X = X +1
5
3
End

Path Coverage

- 路径覆盖定义：

Designing test cases as many as possible and running them so that all paths in program are executed.

All paths:

1-4-5

1-4-3

1-2-3

1-2-5

All paths:

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

test cases:

A=1,B=1,X=1, path 1-2-3

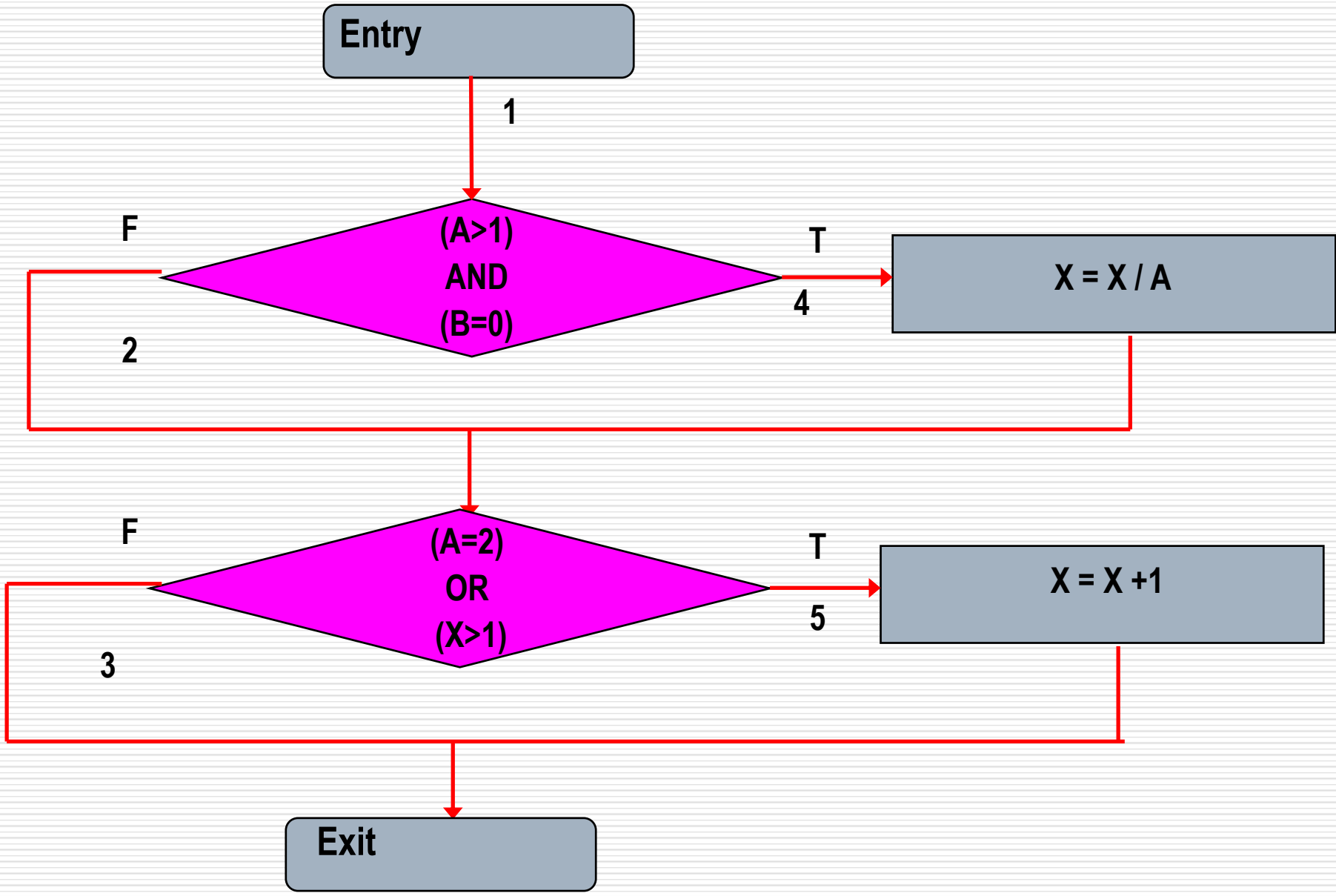
A=1,B=1,X=2, path 1-2-5

A=3,B=0,X=1, path 1-4-3

A=2,B=0,X=4, path 1-4-5

conclusions:

path coverage + condition combination coverage is the strongest testing


Entry
1
(A>1)
AND
(B=0)
F
T
2
4
X = X / A
(A=2)
OR
(X>1)
F
T
3
5
X = X +1
Exit

Point Coverage（点覆盖）

It is equal to statement coverage

Edge Coverage（边覆盖）

It is equal to branch coverage