SOFTWARE TESTING

苏临之
sulinzhi029@nwu.edu.cn

Four Basic Techniques

- Static Black-Box Testing
- Dynamic Black-Box Testing
- Static White-Box Testing
- Dynamic White-Box Testing

Four Basic Techniques

- Static Black-Box Testing
- Dynamic Black-Box Testing
- **Static White-Box Testing**
- Dynamic White-Box Testing

Static White-Box Testing

- Static white-box testing is the process of **carefully and methodically reviewing the software design, architecture, or code** without executing it.
- The obvious reason to perform static white-box testing is to find bugs early and to find bugs that would be difficult to uncover or isolate with dynamic black-box testing.

Contents in the Review Checklist

- The list generally contains the follow eight issues.

1. Data Reference Errors
2. Data Declaration Errors
3. Computation Errors
4. Comparison Errors
5. Control Flow Errors
6. Subroutine Parameter Errors
7. Input / Output Errors
8. Other Checks

Four Basic Techniques

- Static Black-Box Testing
- Dynamic Black-Box Testing
- Static White-Box Testing
- **Dynamic White-Box Testing**

Dynamic White-Box Testing

- Based on the programming code, the dynamic white-box testing is also called code-based testing.
- The basic idea of DWBT is that every individual part of code should be operated once at least, which means to design test cases based on control flows and then to analysis the logic in the code. Finally, the test cases are applied and check the output results.

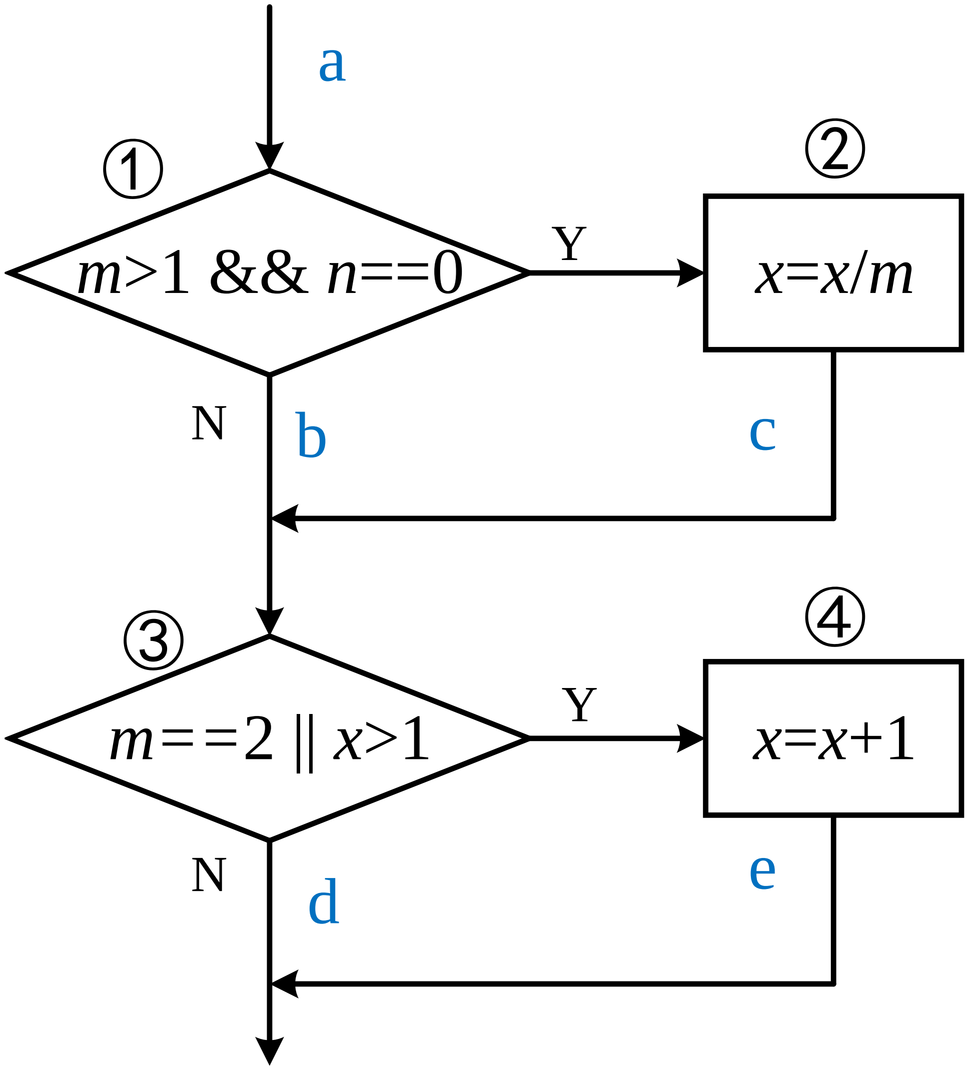
Example 1

- Please design test cases by using the dynamic white-box testing techniques for this piece of code below.

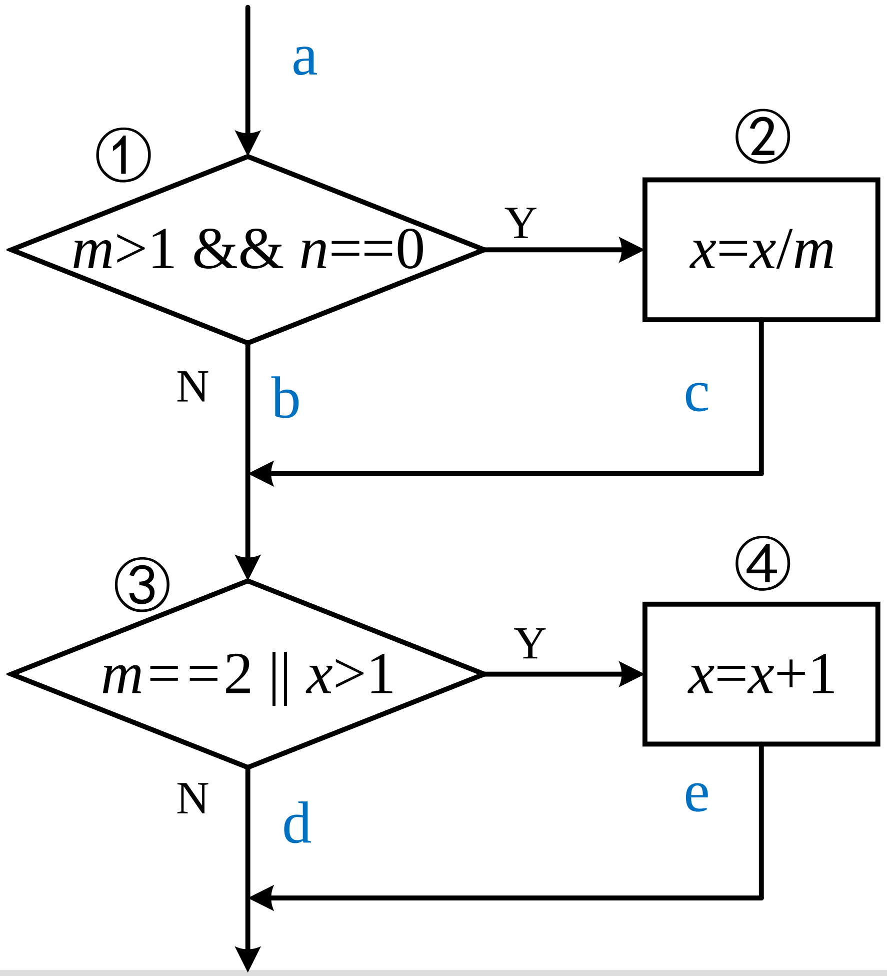
```
if( m>1&&n == 0){
    x = x/m;
}
if( m == 2 || x >1 ){
    x = x + 1;
}
```

Program Chart

```
if( m>1&&n == 0){
    x = x/m;
}
if( m == 2 || x >1 ){
    x = x + 1;
}
```

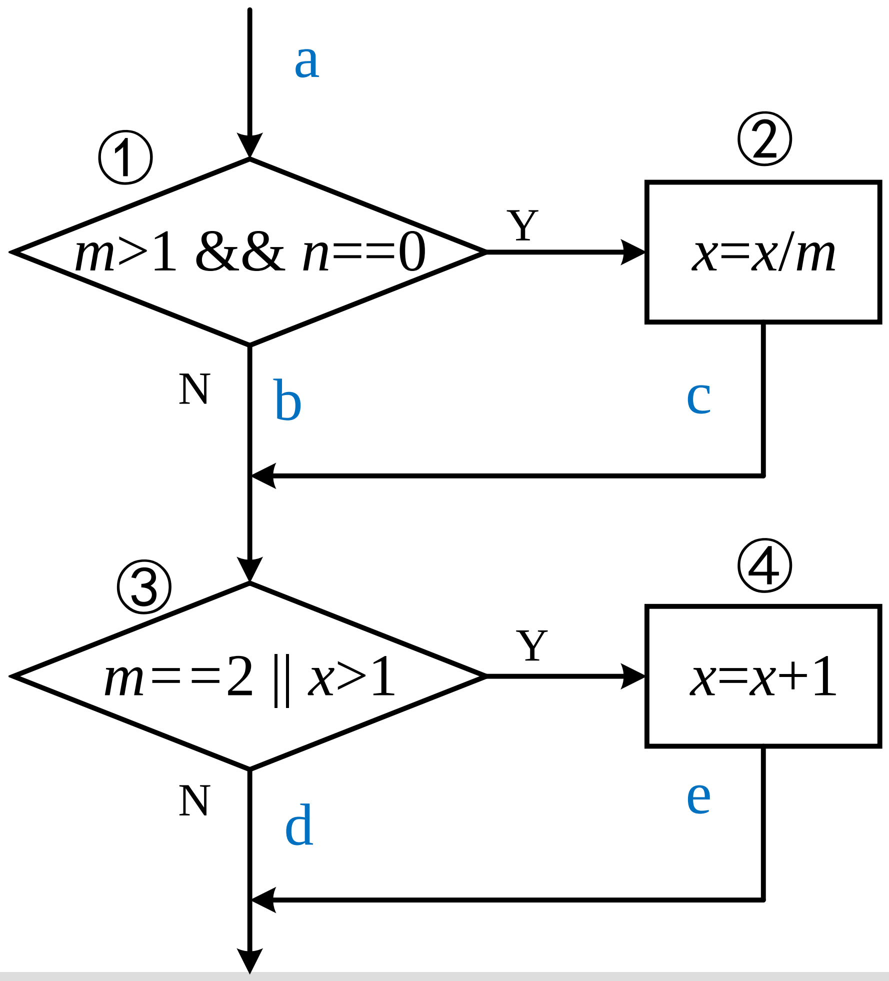


Statement and Path



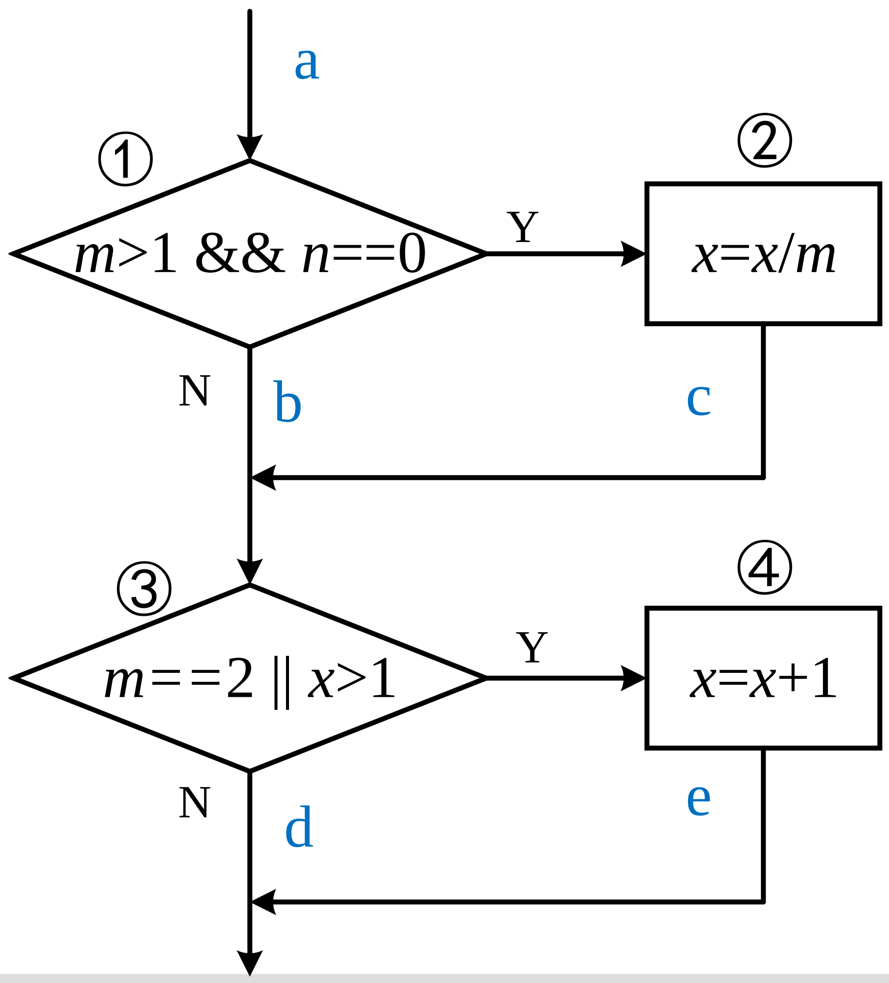
- In the chart, every block represents a line of code, called a **statement (语句)**. Every possible road along which data flow from the start to the end is a **path (路径)**.

Statement and Path



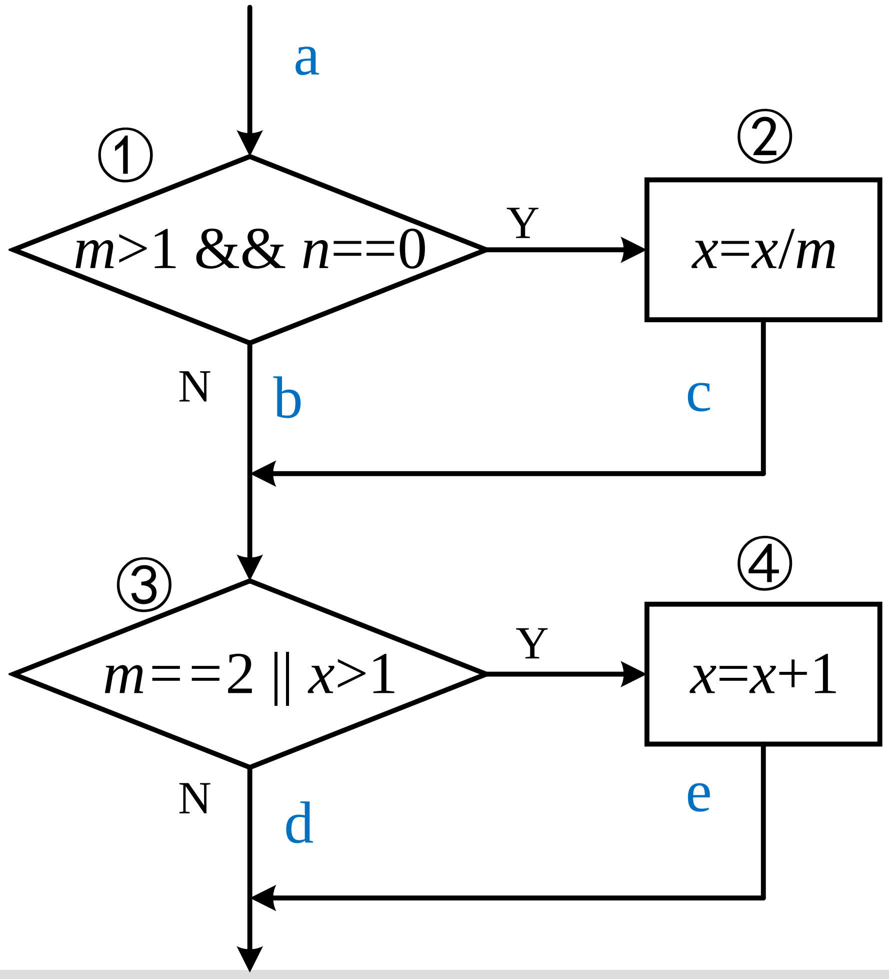
- Four statements can be found ①②③④.
- Four possible paths can be found: abd, acd, abe & ace.

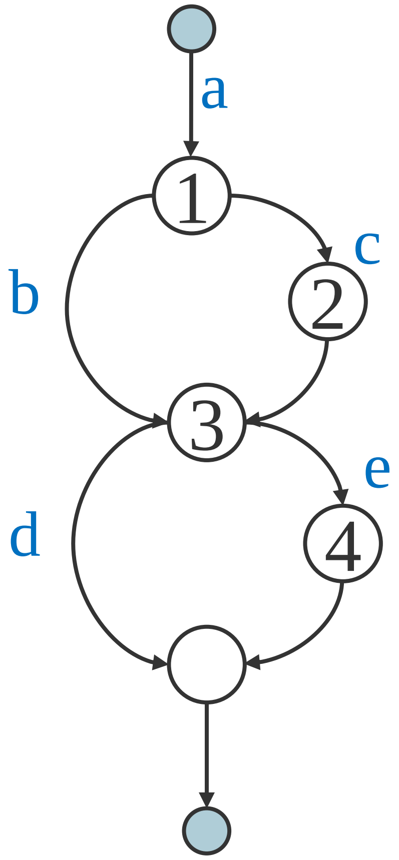
Condition and Decision



- Every diamond box is an IF-ELSE statement, whose sub-term is a **condition (条件)**. The branch (Y & N) is the corresponding **decision (判定分支)**.

Flow Graph



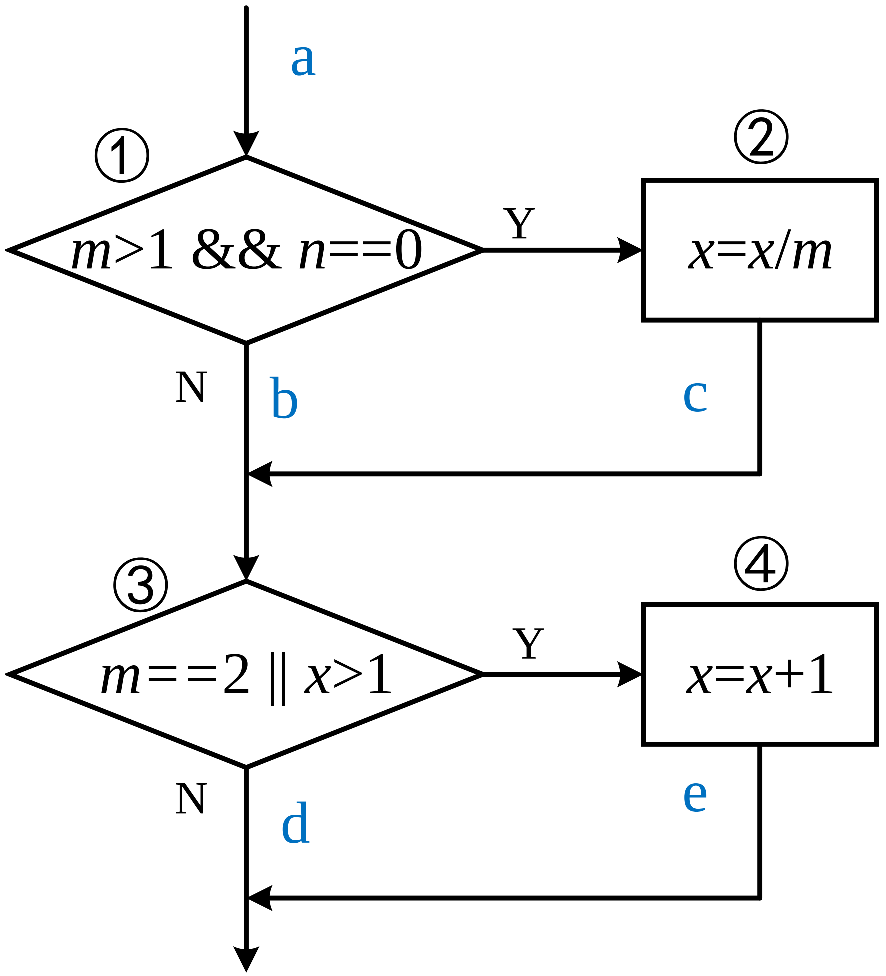


Techniques in DBWT

- Statement Coverage (语句覆盖)
- Decision Coverage (判定覆盖)
- Condition Coverage (条件覆盖)
- Decision / Condition Coverage (判定/条件覆盖)
- Multiple Condition Coverage (条件组合覆盖)
- Path Coverage (决策路径覆盖)

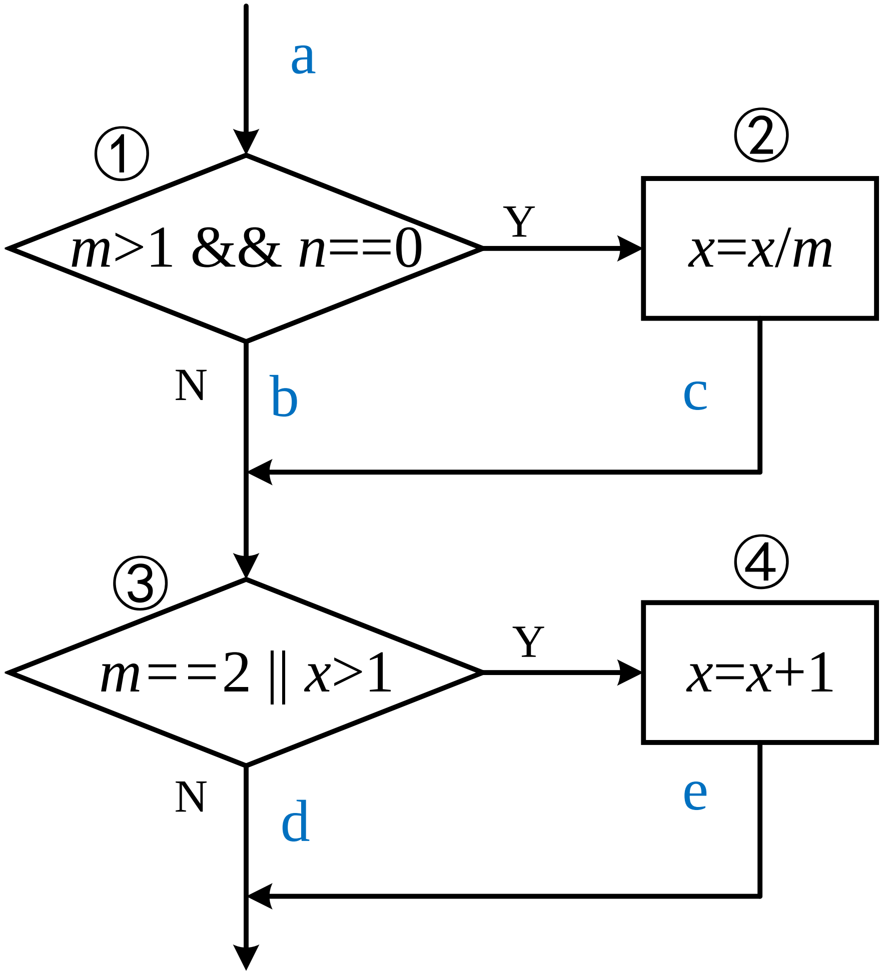
Statement Coverage

- In the statement coverage, one group of test cases will lead to the execution of as many statement as possible.



Statement Coverage

- In the statement coverage, one group of test cases will lead to the execution of as many statement as possible.
- We are able to execute all the statements of ①②③④ by choosing the path ace. Therefore, only one test case will be designed.

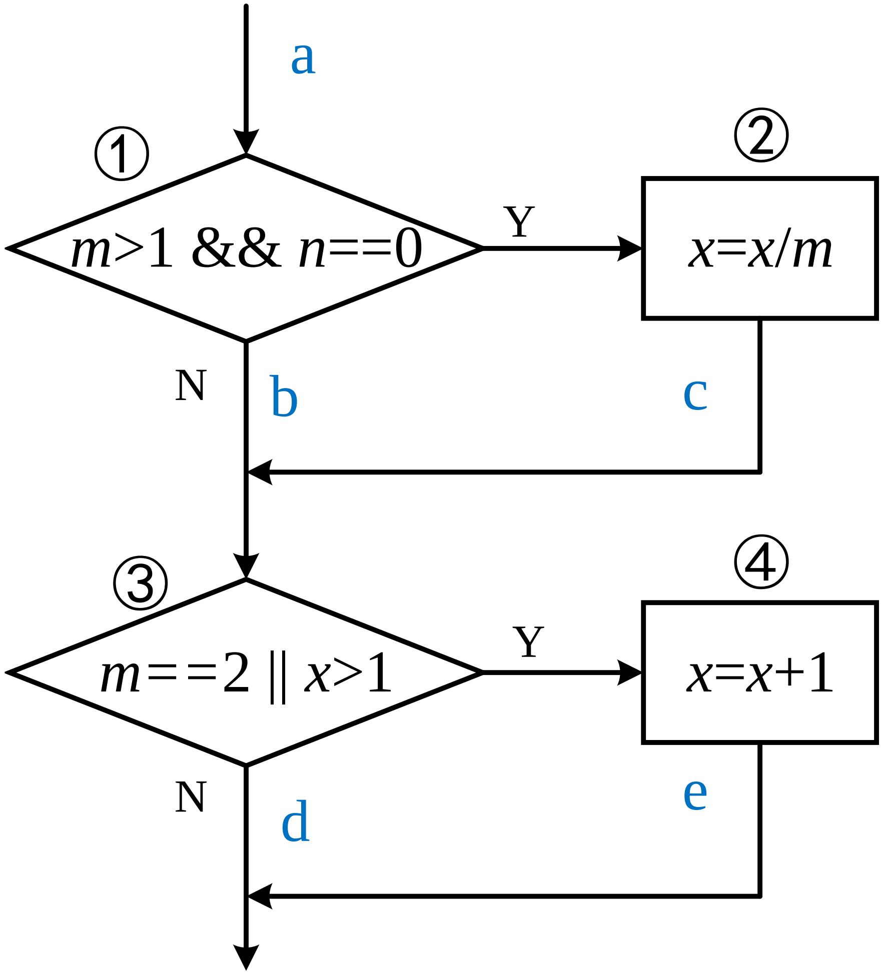


Statement Coverage

用例编号	测试用例 $(m/n/x)$	覆盖语句	预期结果	实际结果
TC1	2,0,4	①②③④	$x = 3$	

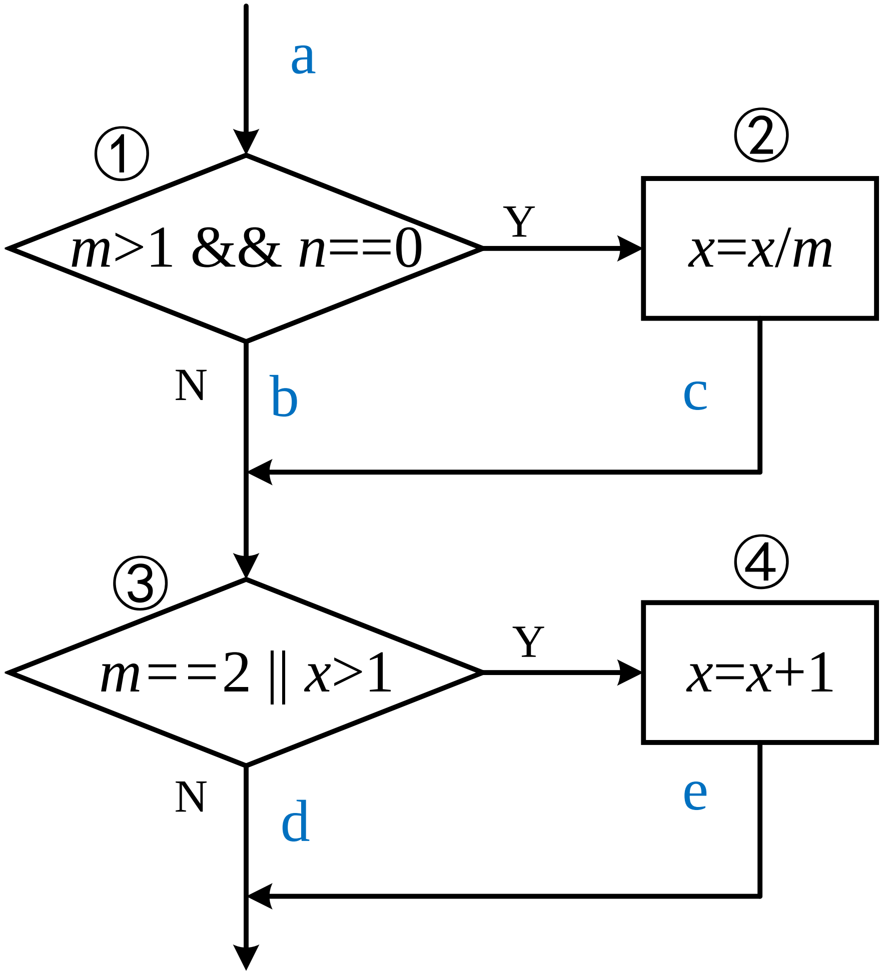
Statement Coverage

- However, it has some obvious disadvantages. For example, if || in ③ is mistaken as &&, (2,0,4) will also cover the path ace. Hence, it is **weakest** of all the coverage techniques and is not able to find some logical errors.



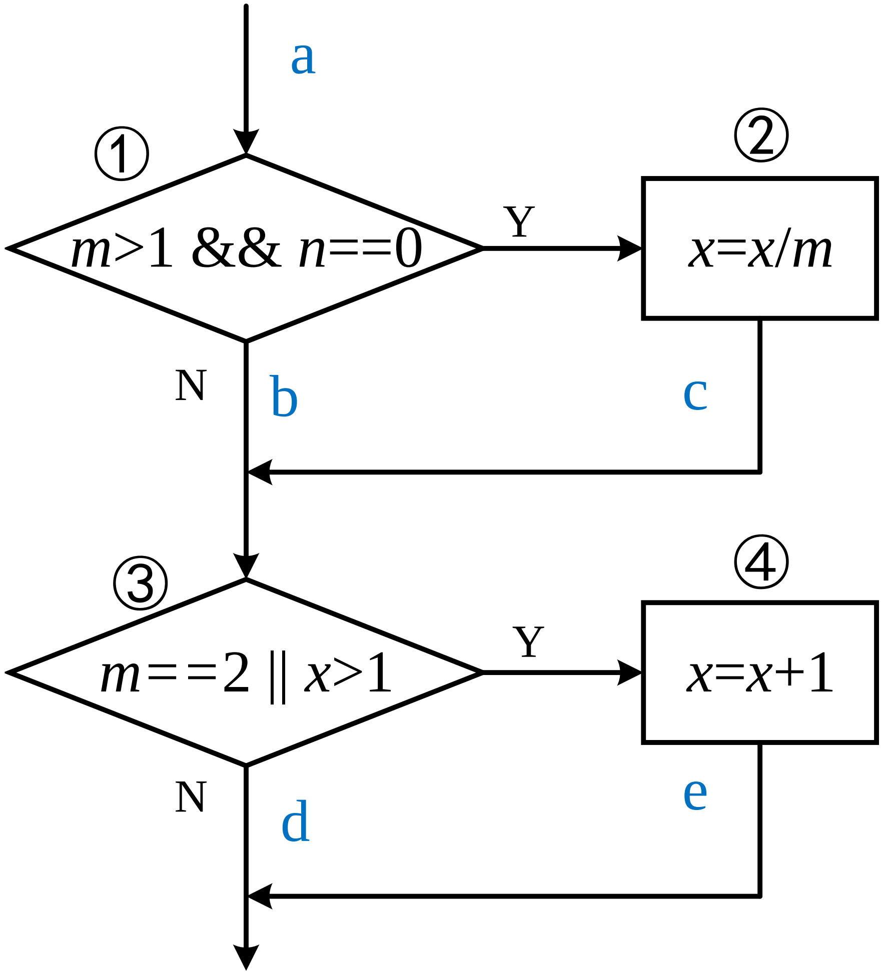
Decision Coverage

- In the decision coverage, the test cases will guarantee the executions of every Y and N. So it is also called branch coverage.



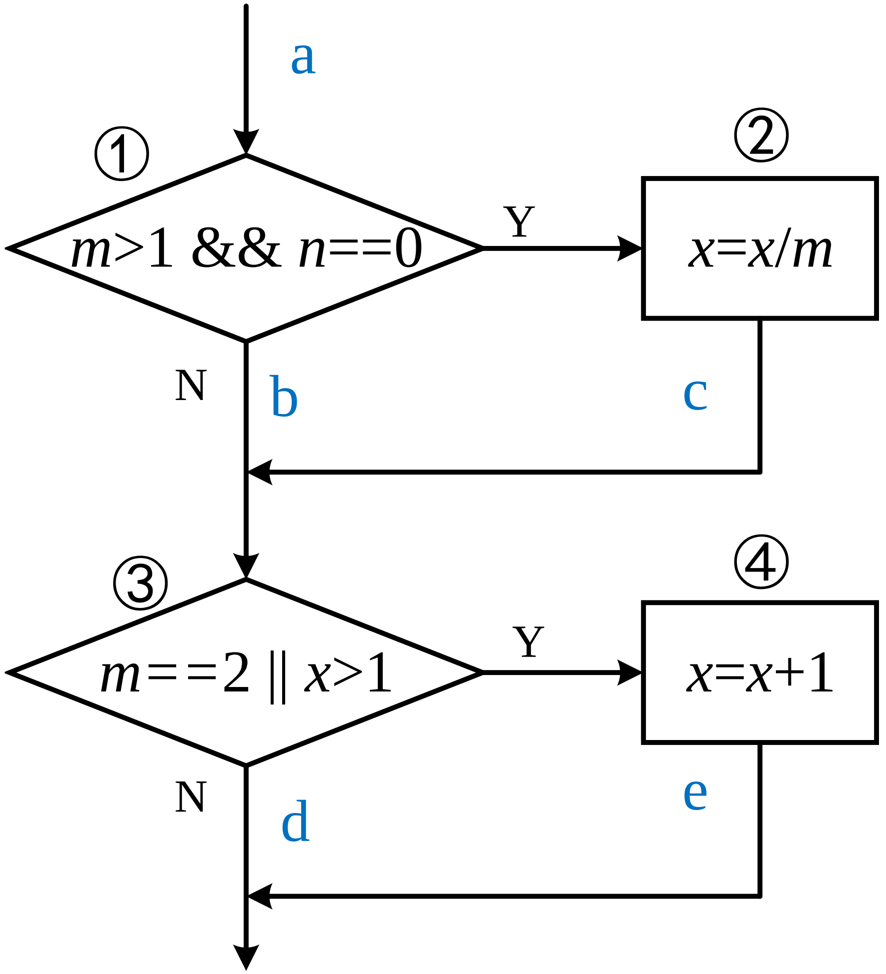
Decision Coverage

- For example, (2,0,4) covers the Ys in ① and ③, while (3,1,1) covers the Ns. Hence, these two test cases can be utilized for decision coverage.



Decision Coverage

- For another example, (3,0,3) covers the Y in ① and the N in③, while (2,1,1) covers the N in ① and Y in ③, so these two test cases can also be utilized for decision coverage.



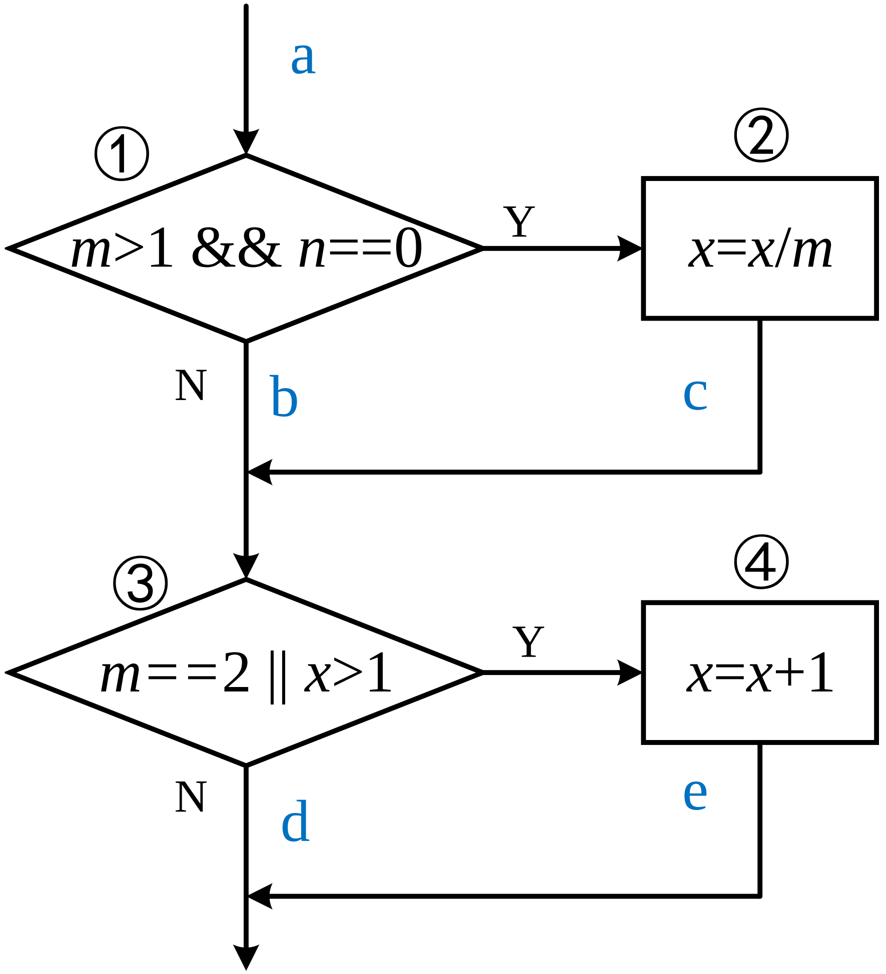
Decision Coverage

用例编号	测试用例 $(m/n/x)$	覆盖分支	预期结果	实际结果
TC1	2,0,4	ce	$x = 3$	
TC2	3,1,1	bd	$x = 1$	

用例编号	测试用例 $(m/n/x)$	覆盖分支	预期结果	实际结果
TC1'	3,0,3	cd	$x = 1$	
TC2'	2,1,1	be	$x = 2$	

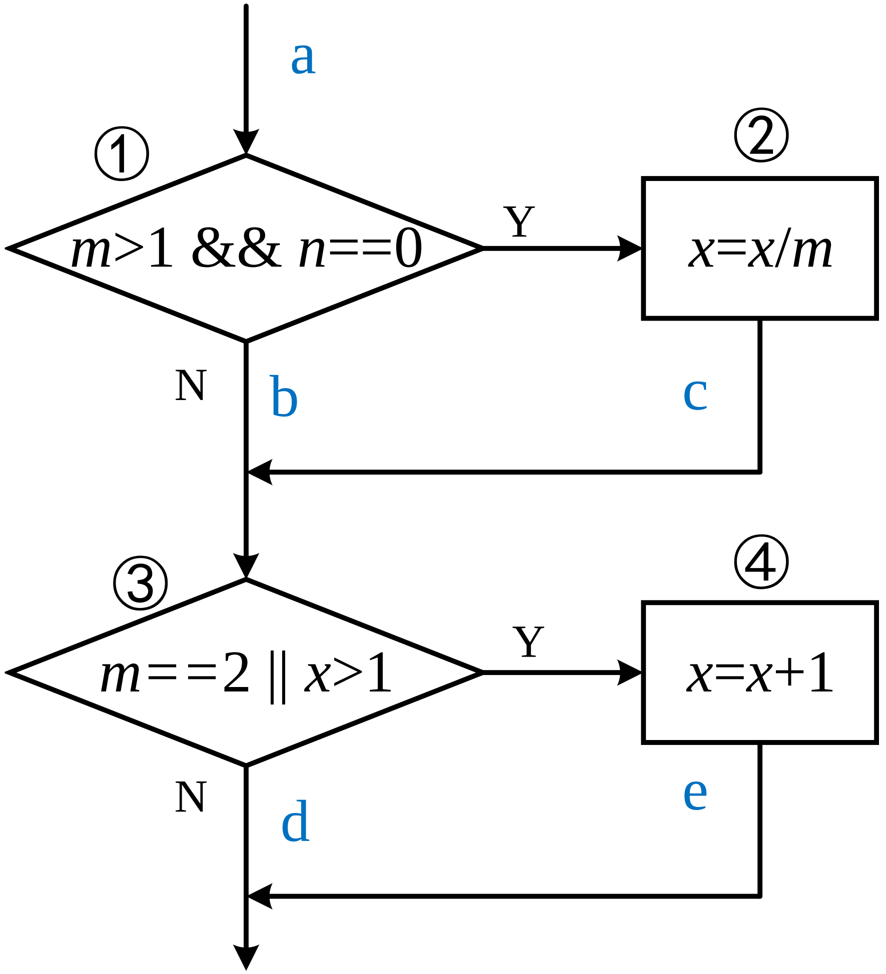
Decision Coverage

- Then how to find the test cases? Actually, the test cases in the statement coverage may enlighten us. Here, (2,0,4) is the one covering two Ys, and by modifying it into (3,1,1), two Ns can be covered.



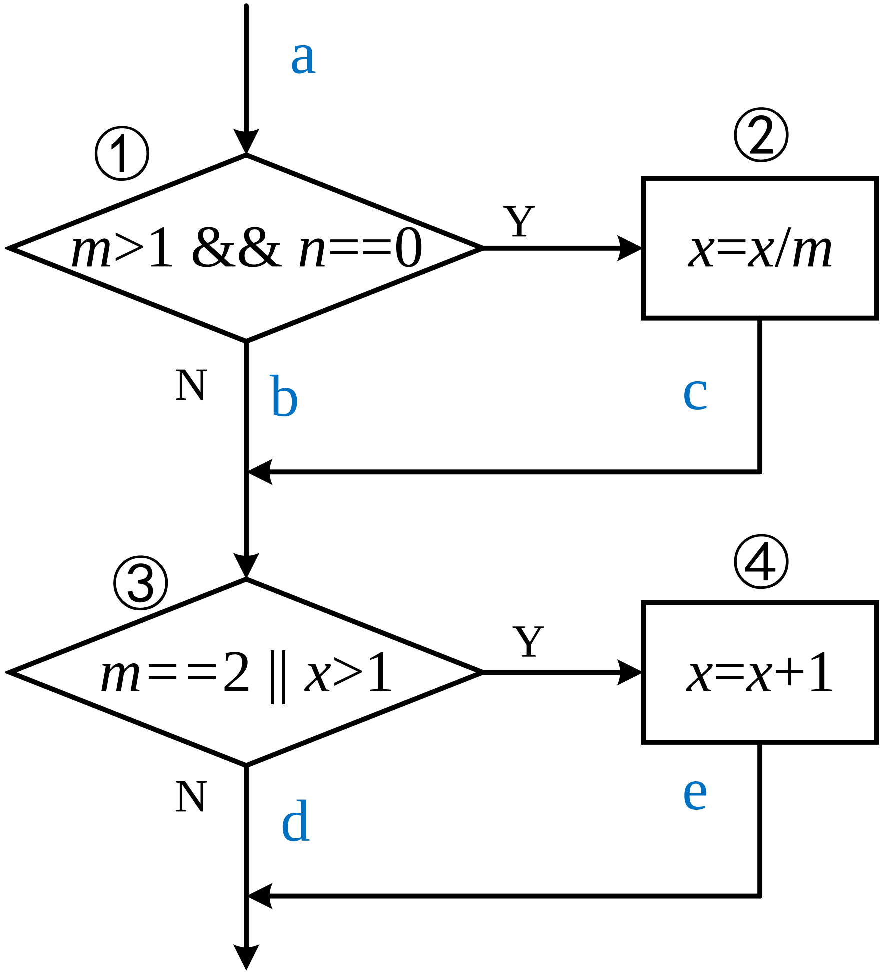
Decision Coverage

- From the process above, the decision coverage is stronger than the statement coverage, but is still not strong enough. For example, if m>1 in ① is mistaken as m>=1, the test cases will be unable to find the errors.



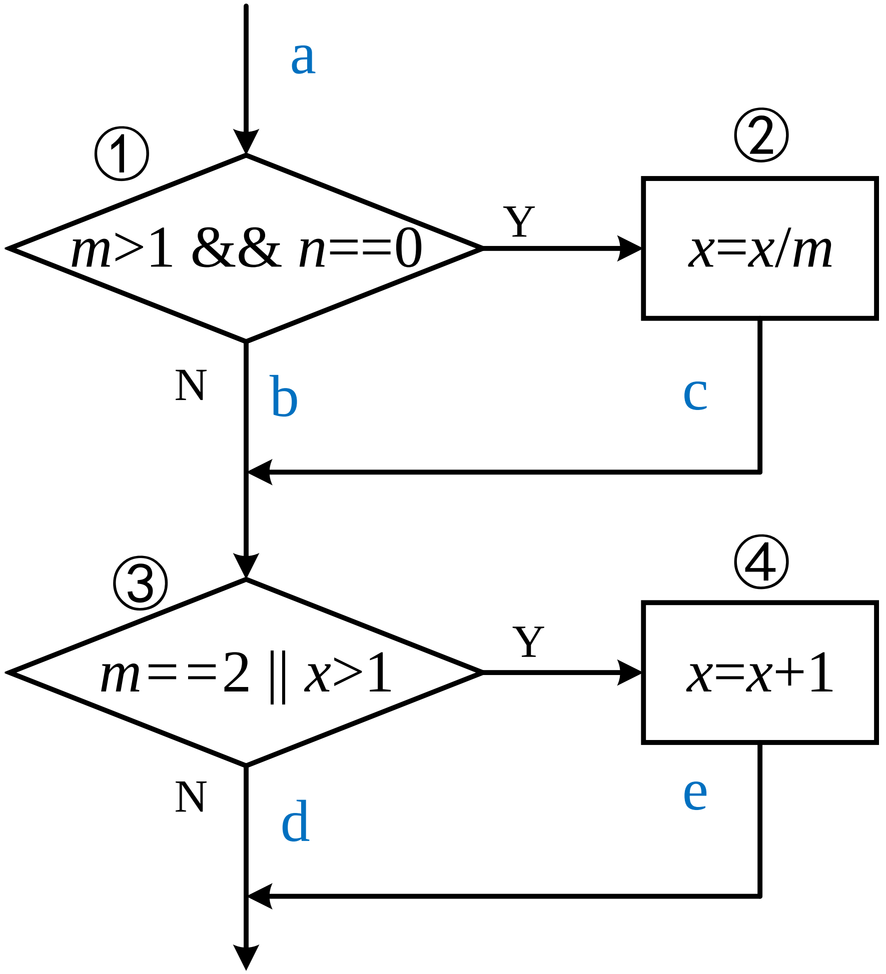
Condition Coverage

- In the condition coverage, every T and F for each condition will be covered.
- Here are four conditions and every condition can value T and F. Therefore, the test cases should cover these eight values.



Condition Coverage

条件满足		条件不满足	
T1	m>1	m≤1	F1
T2	n=0	n≠0	F2
T3	m=2	m≠2	F3
T4	x>1	x≤1	F4



Condition Coverage

- Here, (1,0,3) can cover F1T2F3T4, and (2,1,1) can cover T1F2T3F4. So these test cases can be utilized for condition coverage.

条件满足		条件不满足	
T1	m>1	m≤1	F1
T2	n=0	n≠0	F2
T3	m=2	m≠2	F3
T4	x>1	x≤1	F4

Condition Coverage

用例编号	测试用例 (*m*/*n*/*x*)	满足条件	预期结果	实际结果
TC1	1,0,3	F1 T2 F3 T4	$x = 4$	
TC2	2,1,1	T1 F2 T3 F4	$x = 2$	

Relationship between DC & CC

用例编号	测试用例 (*m*/*n*/*x*)	满足条件	覆盖分支	通过路径
TCC1	1,0,3	F1 T2 F3 T4	be	abe
TCC2	2,1,1	T1 F2 T3 F4	be	abe
TCD1	2,0,4	T1 T2 T3 T4	ce	ace
TCD2	3,1,1	T1 F2 F3 F4	bd	abd

Relationship between DC & CC

用例编号	测试用例 (*m*/*n*/*x*)	满足条件	覆盖分支	通过路径
TCC1	1,0,3	F1 T2 F3 T4	be	abe
TCC2	2,1,1	T1 F2 T3 F4	be	abe
TCD1	2,0,4	T1 T2 T3 T4	ce	ace
TCD2	3,1,1	T1 F2 F3 F4	bd	abd

Relationship between DC & CC

用例编号	测试用例 (*m*/*n*/*x*)	满足条件	覆盖分支	通过路径
TCC1	1,0,3	F1 T2 F3 T4	be	abe
TCC2	2,1,1	T1 F2 T3 F4	be	abe
TCD1	2,0,4	T1 T2 T3 T4	ce	ace
TCD2	3,1,1	T1 F2 F3 F4	bd	abd

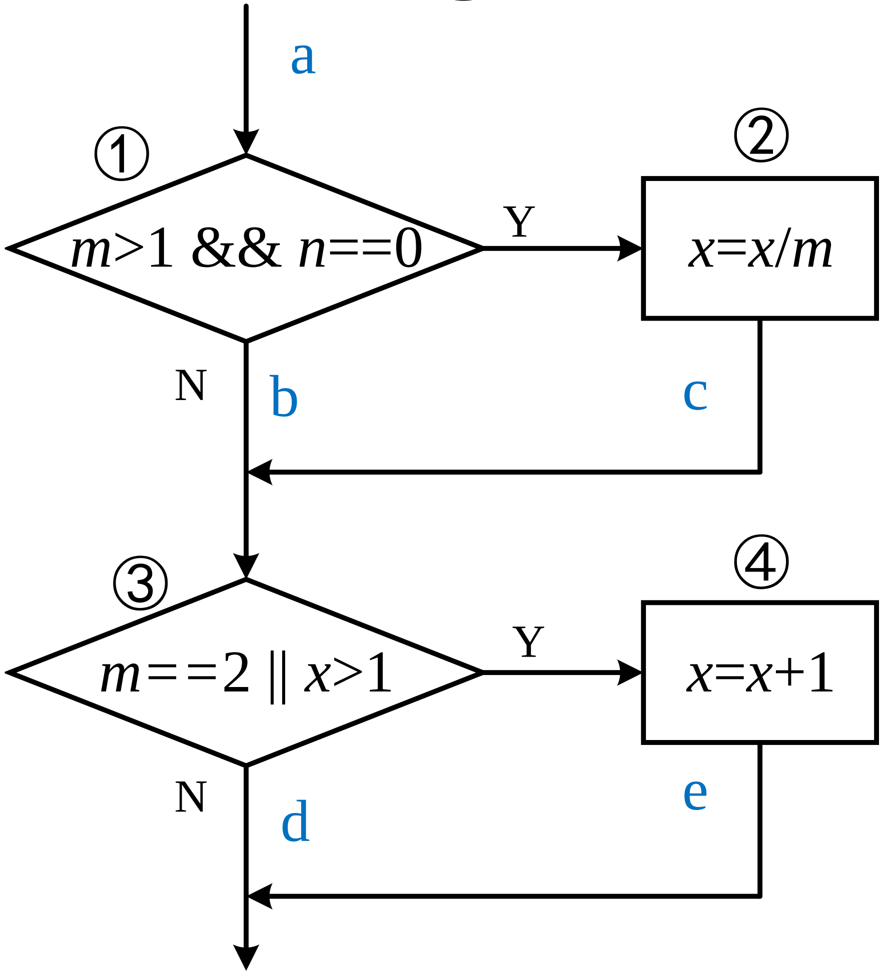
Relationship between DC & CC

用例编号	测试用例 (*m*/*n*/*x*)	满足条件	覆盖分支	通过路径
TCC1	1,0,3	F1 T2 F3 T4	be	abe
TCC2	2,1,1	T1 F2 T3 F4	be	abe
TCD1	2,0,4	T1 T2 T3 T4	ce	ace
TCD2	3,1,1	T1 F2 F3 F4	bd	abd

- Actually they are **neither sufficient nor necessary** to each other.

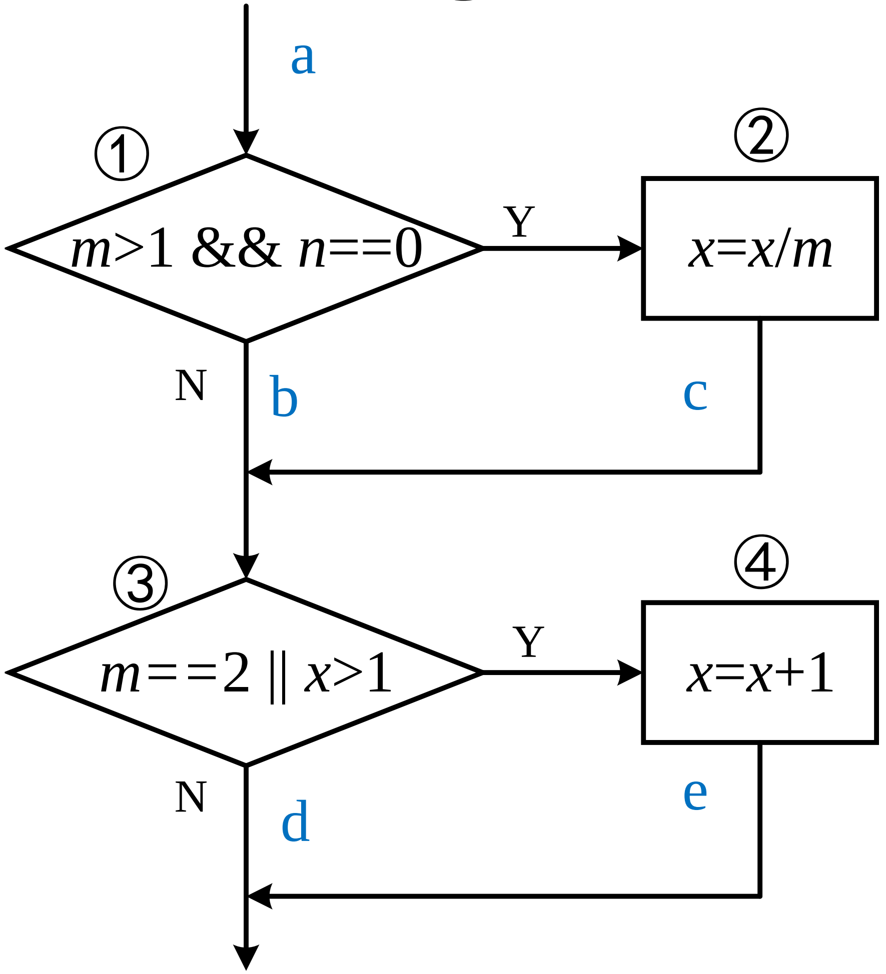
Decision / Condition Coverage

- If the designed test cases can cover every Y and N for decision and every T and F for condition, this is call the decision / condition coverage.



Decision / Condition Coverage

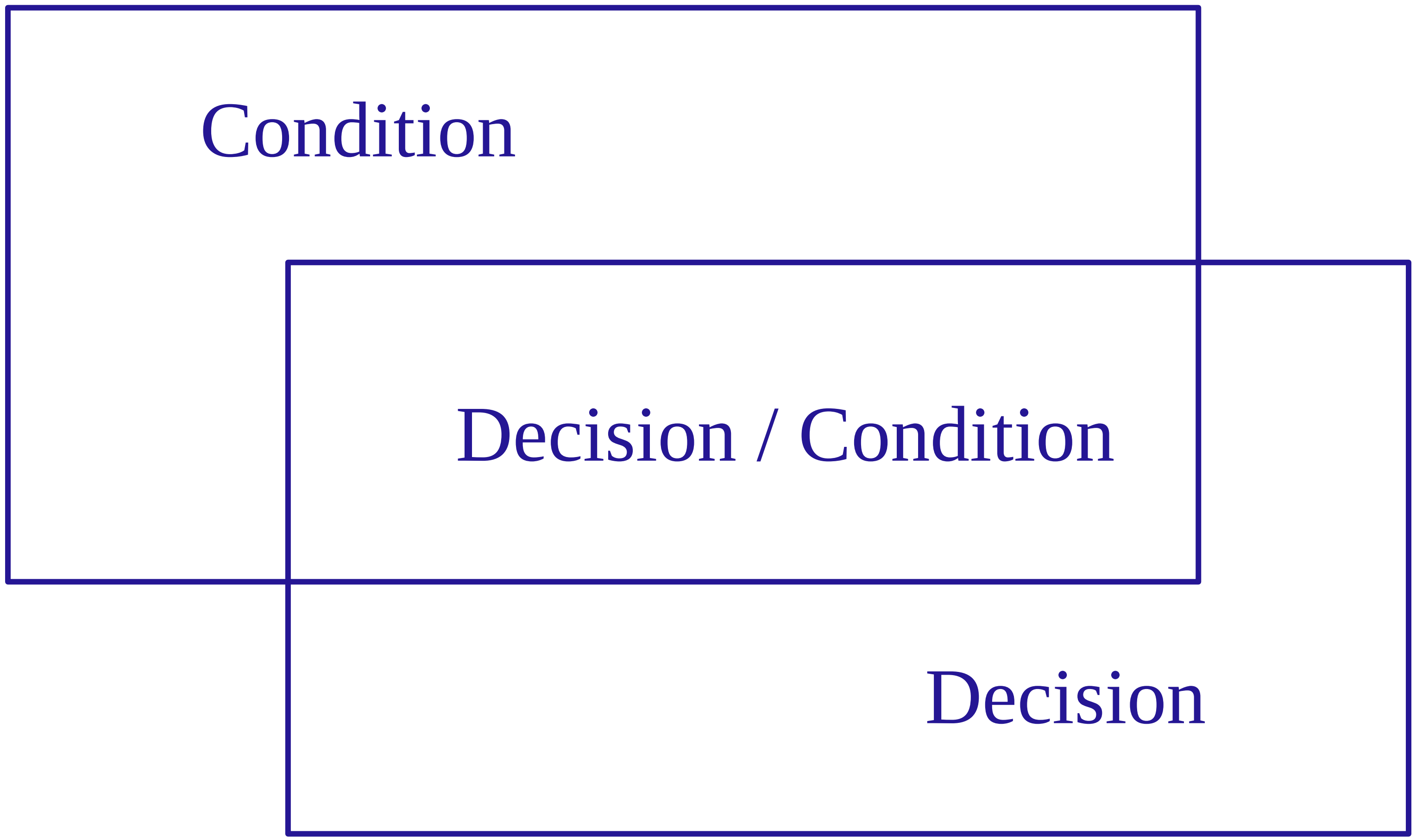
- If the designed test cases can cover every Y and N for decision and every T and F for condition, this is call the decision / condition coverage.
- For example, (2,0,4) and (1,1,1) are the ones that cover all decisions and conditions.



Decision / Condition Coverage

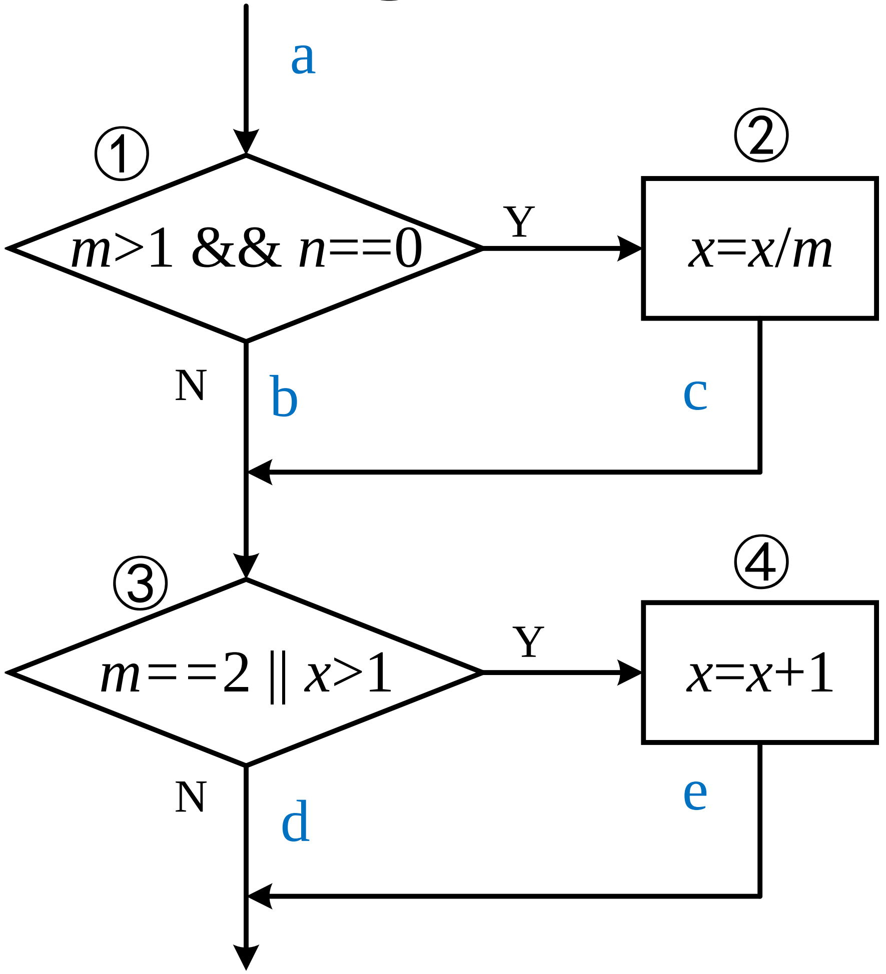
用例编号	测试用例 (*m*/*n*/*x*)	满足条件	覆盖分支	通过路径
TC1	2,0,4	T1 T2 T3 T4	ce	ace
TC2	1,1,1	F1 F2 F3 F4	bd	abd

Relations



Multiple Condition Coverage

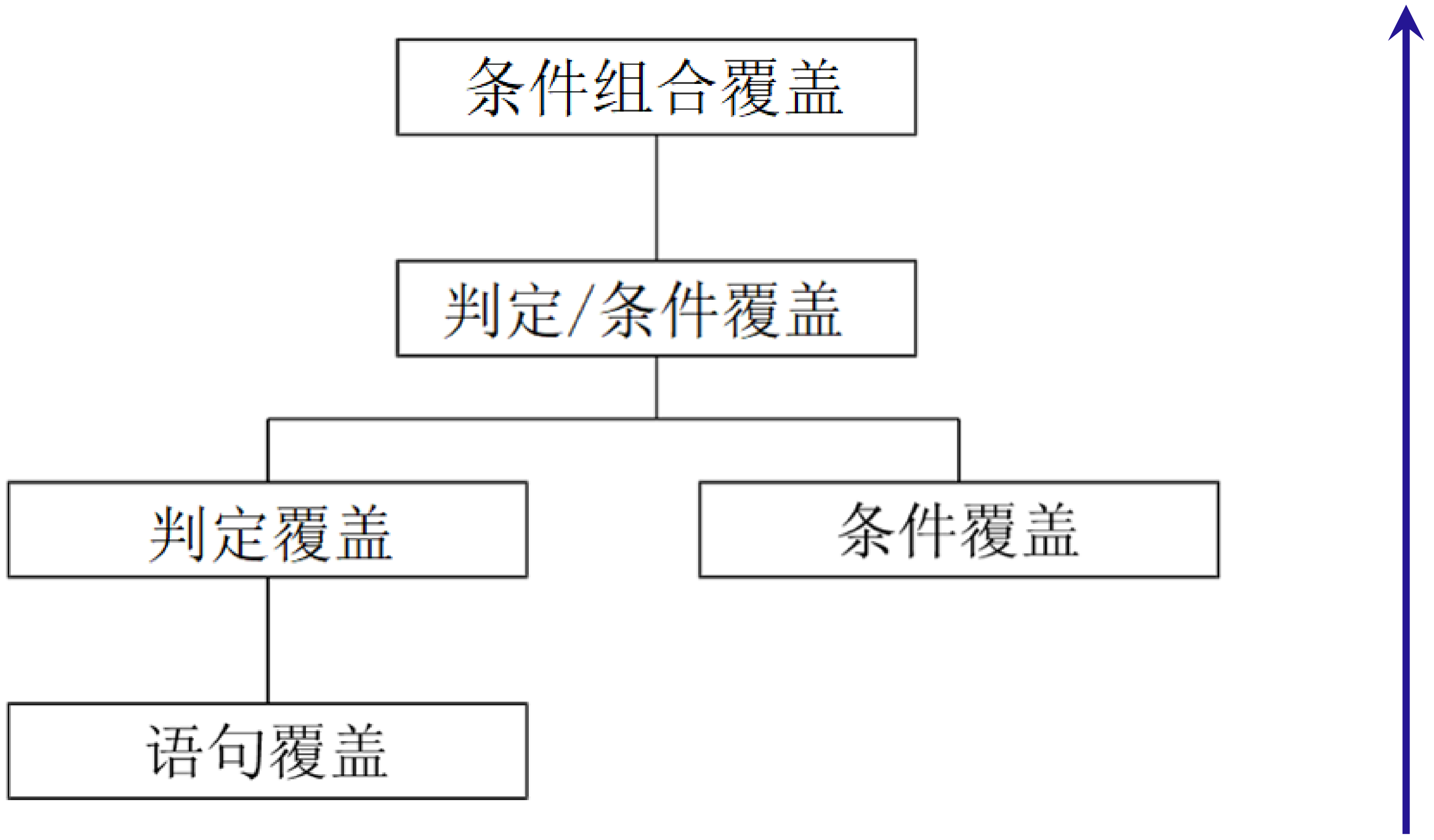
- For ①, four multiple condition combinations are involved: T1T2, T1F2, F1T2 & F1F2. For ③, we also have four combinations: T3T4, T3F4, F3T4 & F3F4.
- In the multiple condition coverage, every combination of Ts and Fs are covered at least once.



Multiple Condition Coverage

用例编号	测试用例 ($m/n/x$)	满足条件组合	预测结果	实际结果
TC1	2,0,4	T1T2, T3T4	$x = 3$	
TC2	2,1,1	T1F2, T3F4	$x = 2$	
TC3	1,0,2	F1T2, F3T4	$x = 3$	
TC4	1,1,1	F1F2, F3F4	$x = 1$	

Analysis of Coverage Degree

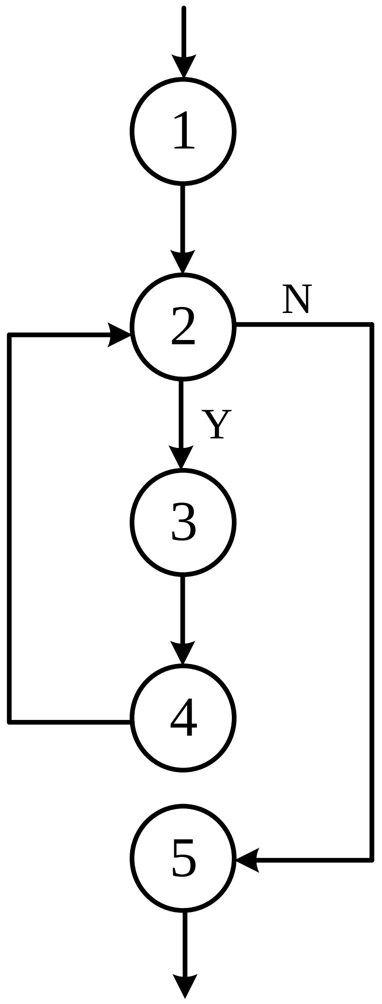


Program Control Flow Graph

- A program, no matter how complicated it is, is made of three basic structure elements: sequence, branch and loop. A program control flow graph (程序控制流图) can be used to describe it.
- In the flow graph, two elements are involved: **node (节点)** and **edge (边)**.

Node

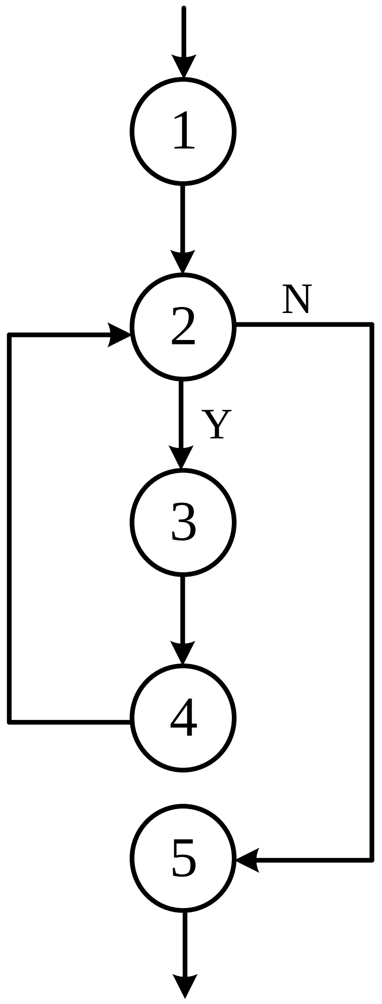
- Each circle in the graph is a node, and if there are several statements in a sequence structure, they are actually viewed as one node.



Node

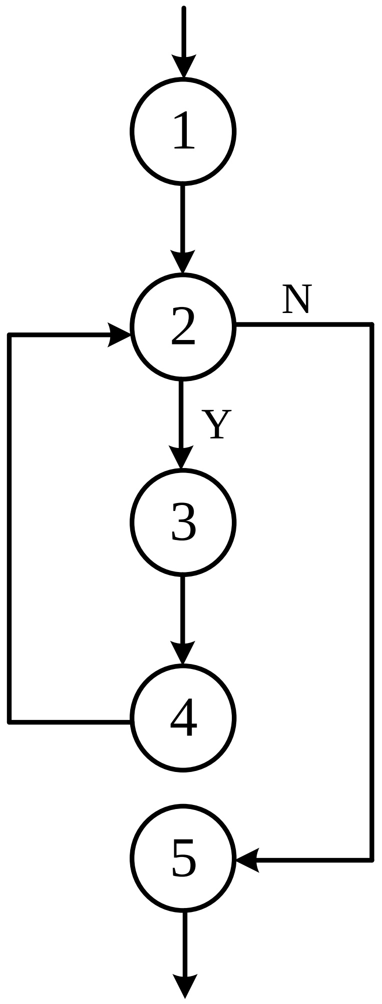
- Each circle in the graph is a node, and if there are several statements in a sequence structure, they are actually viewed as one node.
- For example, the piece of code below, which consists several statements, are actually one node.

```
double pai=3.1416, F=10;
double r1=5, r2=7;
p1=F/(pai*r1*r1);p2=F/(pai*r2*r2);
```



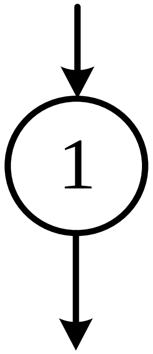
Edge

- An edge shows the sequence of nodes.
- The edge which enter the program is called the entry edge (入口边), and the one which leave the program is called the exit edge (出口边). The other edges are the active edge (有效边).



Sequence Structure

- In the sequence structure, no matter how many statements there are, only one node are included. And it is obvious that one entry edge and one exit edge are involved. Obviously, it contains only one path.

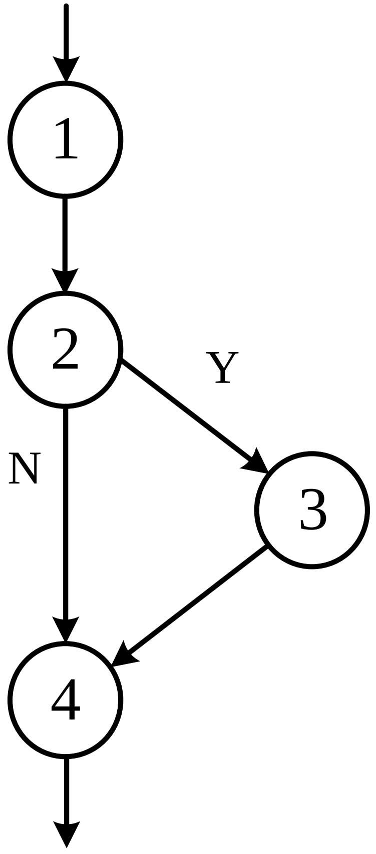


Branch Structure

- The branch structure usually involves the IF statement. When it is a simple IF statement as shown here, two paths are contained.

P1：1-2-4

P2：1-2-3-4

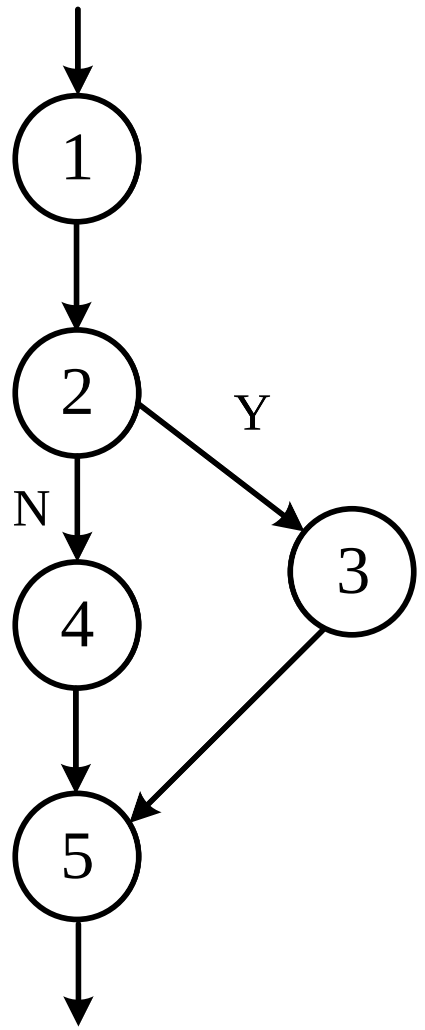


Branch Structure

- When the IF statement contains an ELSE, there are also two paths. And we can find more paths when it comes to some more complicated structures (e.g. SWITCH-CASE).

P1：1-2-3-5

P2：1-2-4-5



Loop Structure

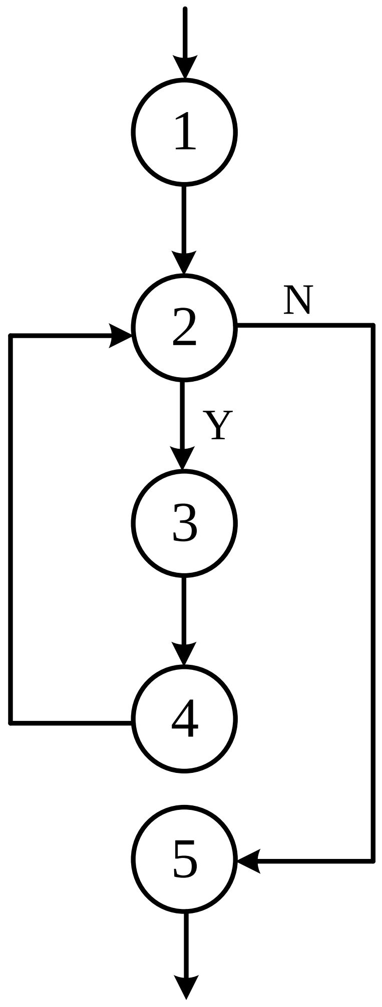
- The FOR and WHILE loop statement involves many paths.

P1：1-2-5

P2：1-2-3-4-2-5

P3：1-2-3-4-2-3-4-2-5

……



Loop Structure

- When the BREAK is involved, some possible paths are shown here.

P1： 1-2-6

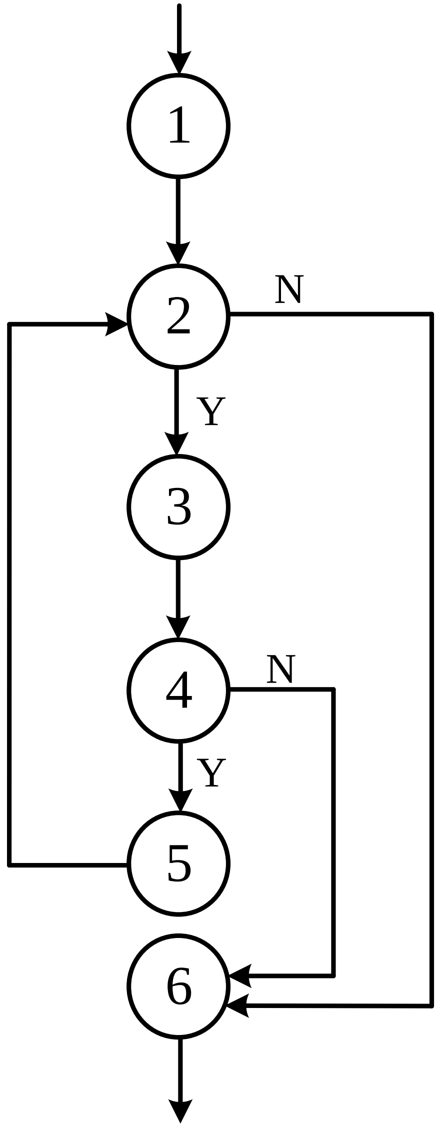
P2： 1-2-3-4-5-2-6

P3： 1-2-3-4-6

P4： 1-2-3-4-5-2-3-4-5-2-6

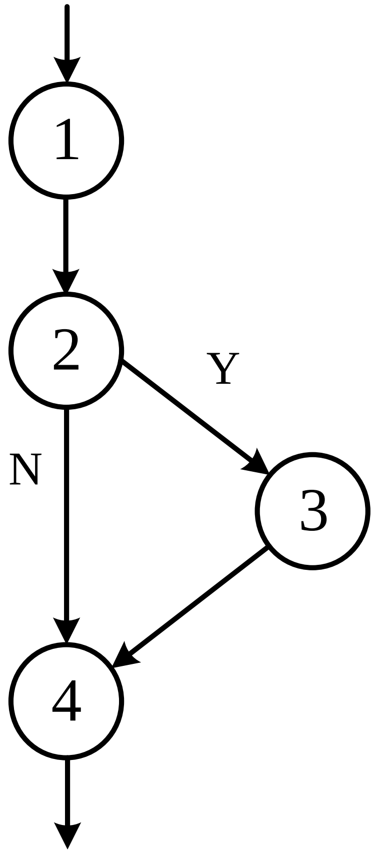
P5： 1-2-3-4-5-2-3-4-6

……



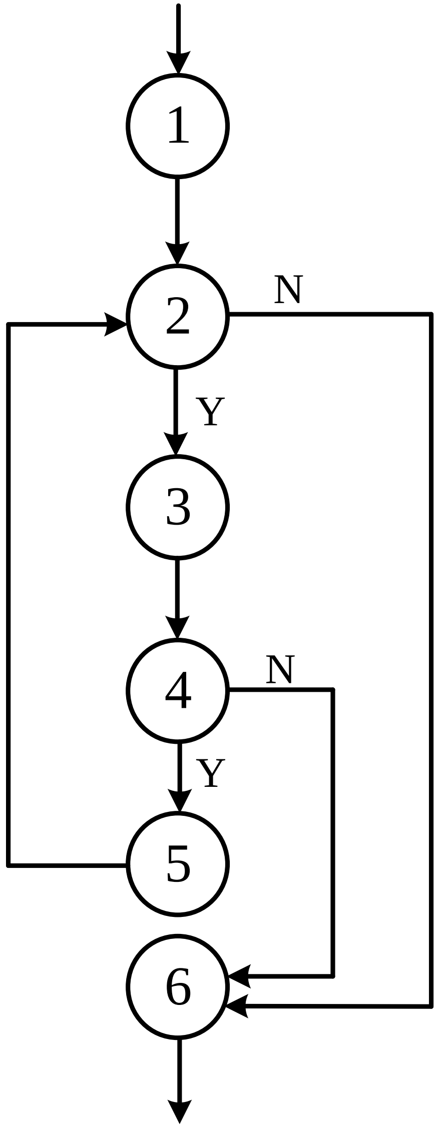
Independent Path

- For the existing paths, if the new path includes at least one edge that is not involved before, then the new path is called an independent path (独立路径).
- For example, the existing path is 1-2-3-4, and then 1-2-4 is its independent path because the edge 2-4 is not involved in 1-2-3-4.



Independent Path

- For another example, if the existing paths are 1-2-6, 1-2-3-4-6 and 1-2-3-4-5-2-6, then the path 1-2-3-4-5-2-3-4-6 is not their independent path because all the edges of the new path have been encountered in the existing paths.



Homework

- 右侧程序输入变量是j、k和y，输出变量是y。请画出框图，然后分别给出语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖和条件组合覆盖的测试用例及预期结果。其中判定覆盖、判定/条件覆盖要指出对应用例覆盖的路径，条件覆盖和判定/条件覆盖要指出对应用例覆盖的条件，条件组合覆盖要指出对应用例覆盖的条件组合。

```
if (j==2||k>2)
    y=y+1;
if (j>3&&y>1)
    y=y+2;
```

- 提示：尽量选用整数以便测试

THANK YOU !