软件质量保证与测试

Software Quality Assurance and Testing

第 4 章 白盒测试

4.3 逻辑覆盖

金陵科技學院

逻辑覆盖

逻辑覆盖是白盒测试中主要的动态测试方法之一，是以程序内部的逻辑结构为基础的测试技术，是通过对程序逻辑结构的遍历来实现对程序的测试覆盖，所谓覆盖就是作为测试标准的逻辑单元、逻辑分支、逻辑取值都被执行到。这一方法要求测试人员对程序的逻辑结构有清楚的了解。

逻辑覆盖的标准有：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖。

逻辑覆盖

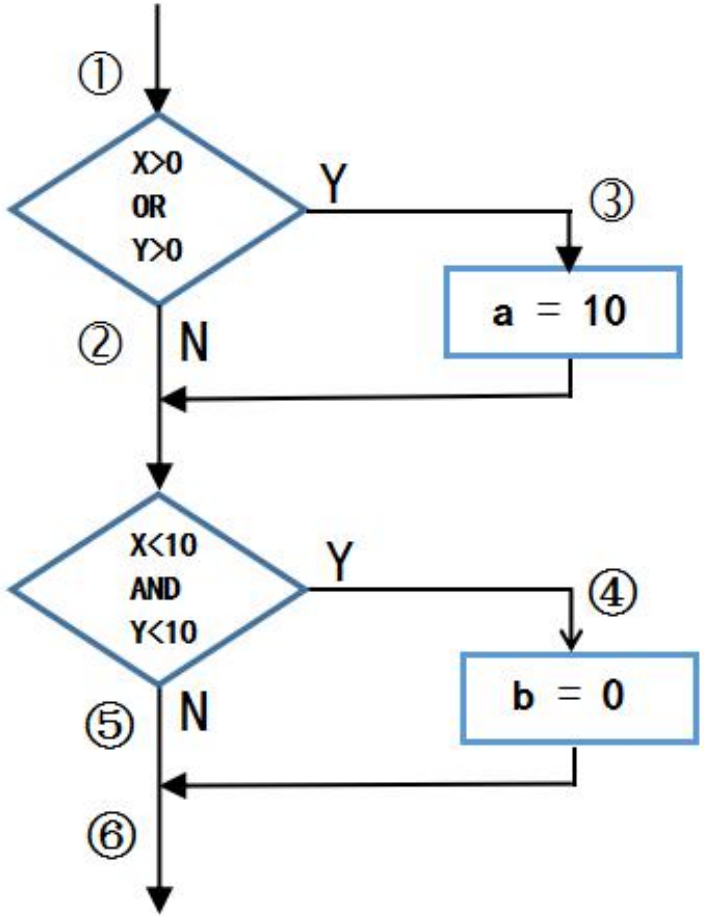
设有如下一段程序 P1：

If （ x>0 OR y>0） then a = 10

If （x<10 AND y<10） then b = 0

其中变量a，b的初始值在其他地方已经定义了，都为 -1。

程序段对应的流程图如图所示。



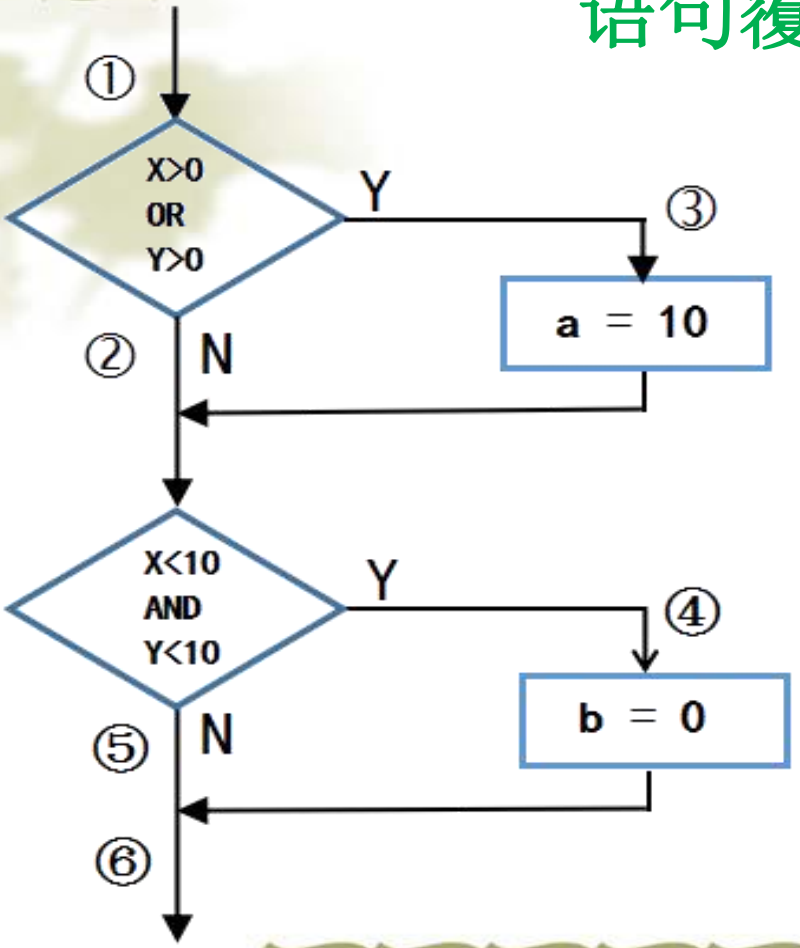
逻辑覆盖

下面我们来看一下应如何来分别实现语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖和条件组合覆盖。

语句覆盖：

语句覆盖要求设计若干个测试用例，使得程序中的每个语句至少都能被执行一次。

语句覆盖



对照流程图，我们来看一下，按照这一标准，程序需要执行通过的位置有①③④⑥，而②⑤位置，由于没有语句，所以不需要覆盖。

语句覆盖

首先我们可能想到的是，可以设计两个测试用例，分别覆盖第一个IF结构有执行语句的分支③，和第二个IF结构有执行语句的分支④，如：

case1：x= 1 ,y= 1,覆盖 ③；
case2：x= -1,y= -1,覆盖 ④。

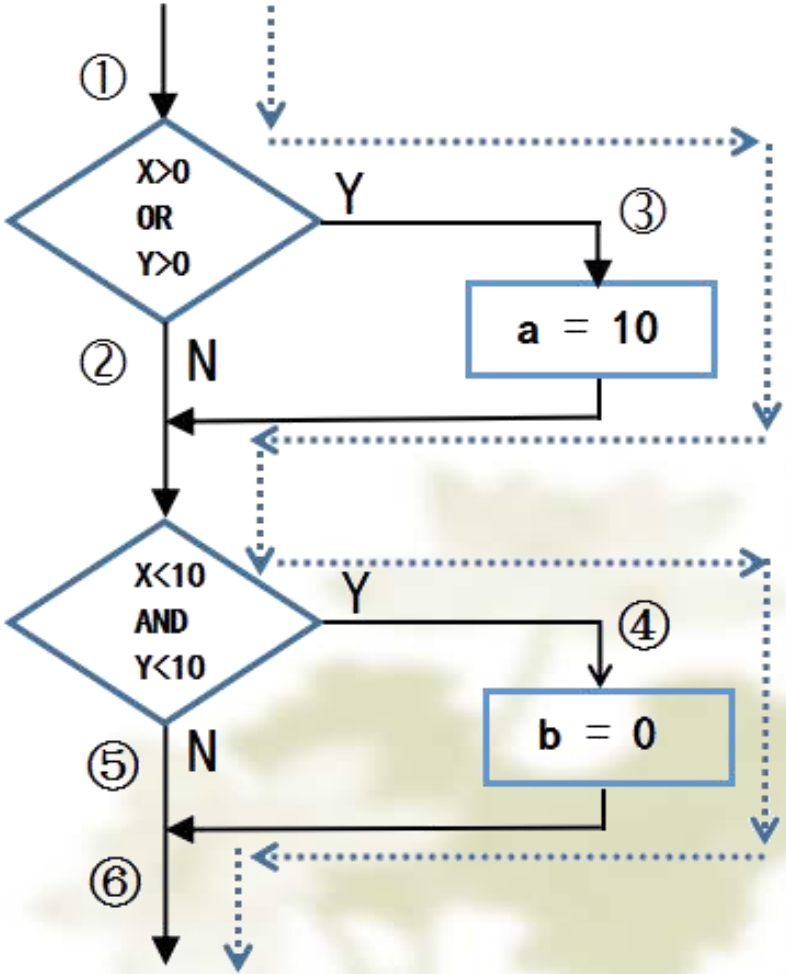
这样即可达到语句覆盖要求。

语句覆盖

但从节约测试成本的角度出发，我们可以优化一下测试用例设计，实际上只需要一个测试用例，如：

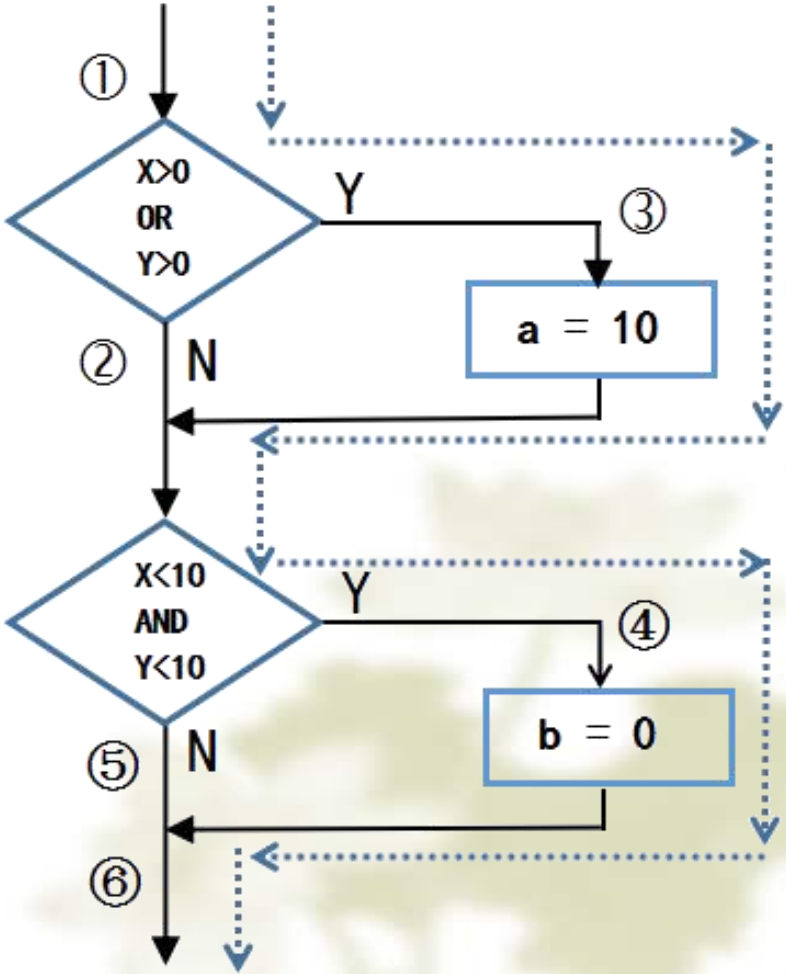
case3：x=8, y=8。

即可同时覆盖①③④⑥，执行通过路径如图所示。



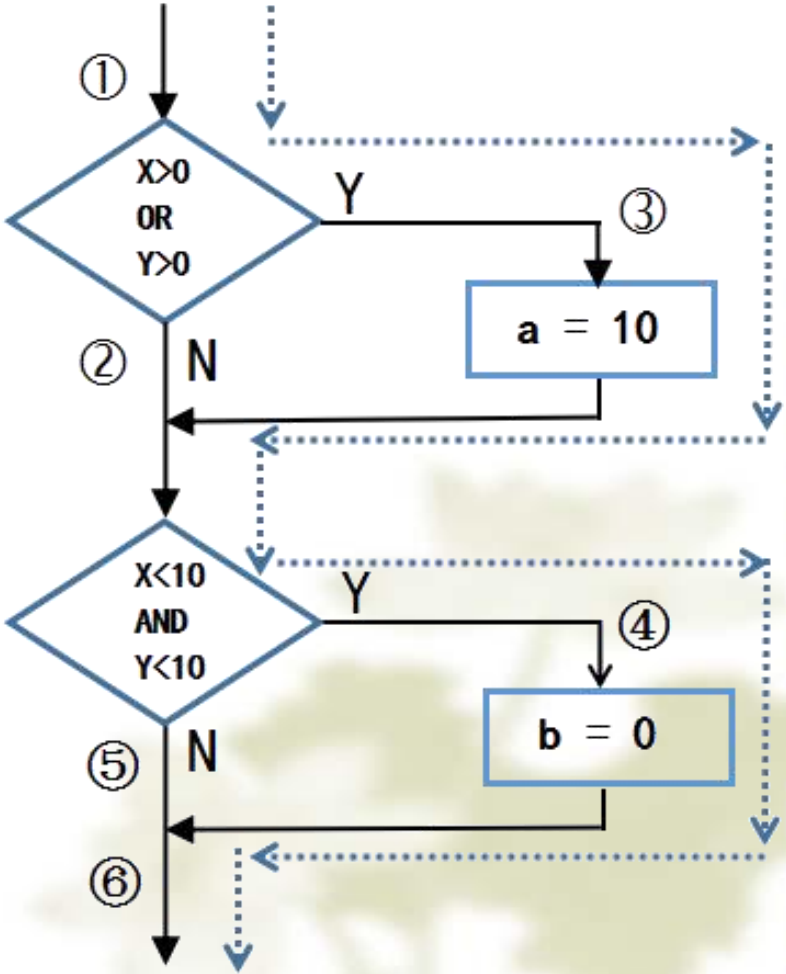
语句覆盖

语句覆盖测试执行了程序中的每一个语句，似乎能够比较全面的对程序进行检验，但实际上，它并不是一个测试很充分的覆盖标准，从图中可以看出，两个判断语句的都只执行了一个分支，而另外一个分支根本就没有被执行到。



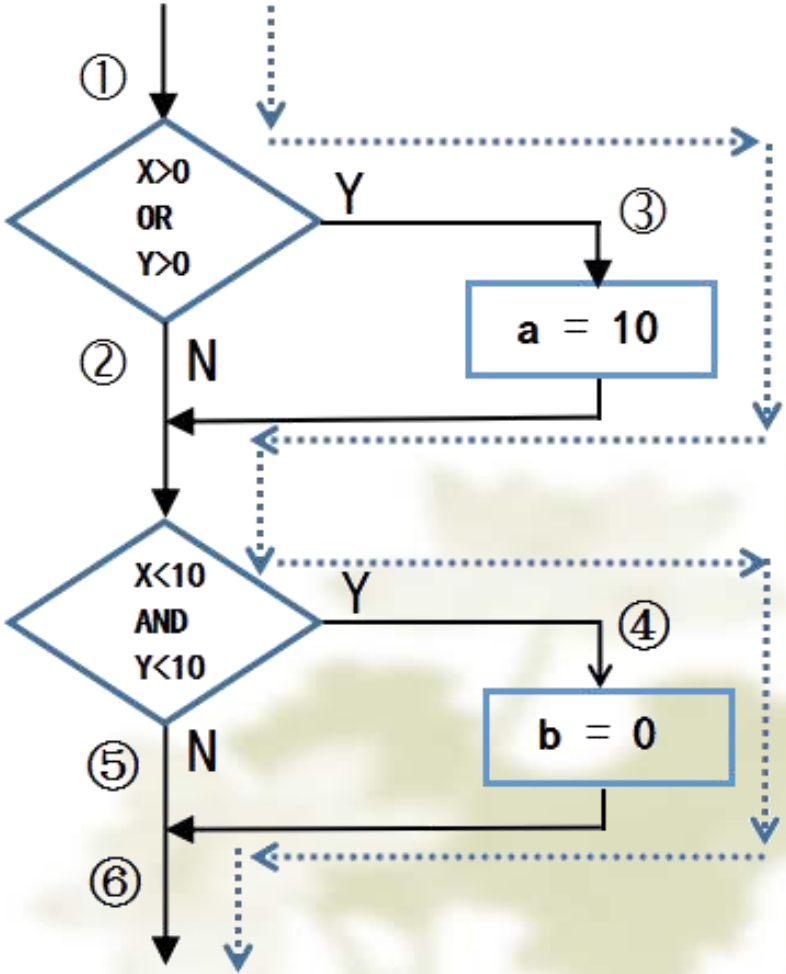
语句覆盖

假设这一程序段中，两个判断语句的逻辑运算符号由于疏忽写错了，第一个判断语句中的“and”错写成了“or”，第二个判断语句中的“or”错写成了“and”，我们用测试用例case3来进行测试，则执行的路径仍然是①③④⑥，测试结果也依然正确，测试没有能够发现程序中的错误。



语句覆盖

语句覆盖，说起来是每一个语句都执行过了，似乎十分可靠，但实际上语句覆盖是一种比较弱的覆盖准则。



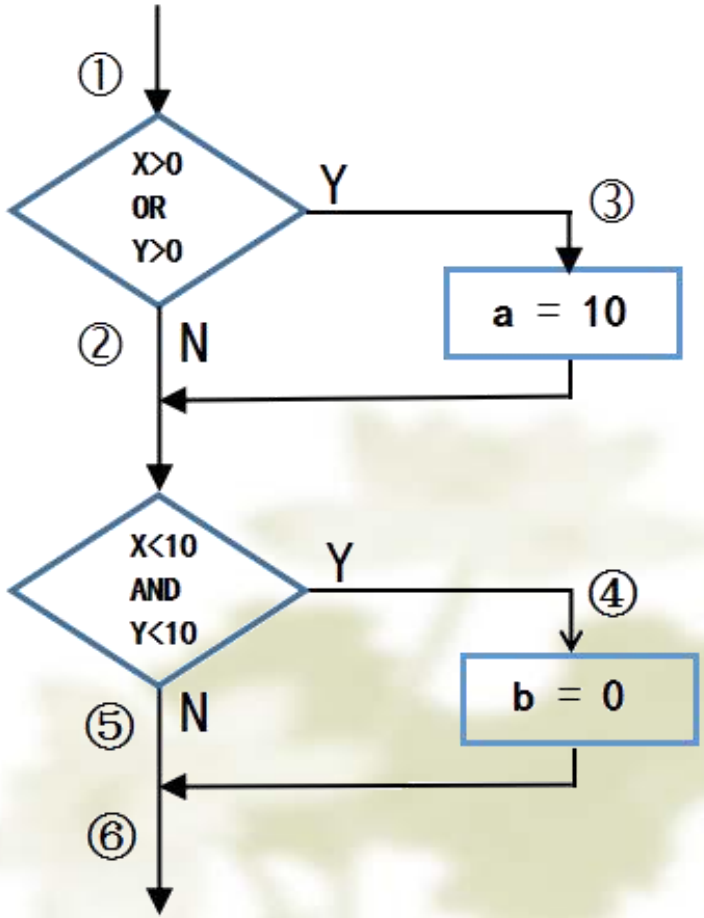
判定覆盖

比语句覆盖稍强的覆盖标准是判定覆盖。判定覆盖是指，设计若干测试用例，运行被测程序，使得程序中每个判断的真值结果和假值结果都至少出现一次。

判定覆盖又称为分支覆盖，因为判断结果取真值就会执行取真分支，判断结果取假值就会执行取假分支，每个判断的真值结果和假值结果都至少出现一次，也就相当于每个判断的取真分支和取假分支至少都经历一次。

判定覆盖

仍以前面的程序段P1为例，对照流程图，按照判定覆盖标准，程序需要执行通过的位置有①②③④⑤⑥。



判定覆盖

程序段P1中存在IF语句，由于每个判断有真假两种判断结果，所以至少需要两个测试用例。P1中的两个IF语句是串联的，不是嵌套，所以如果设计合理的话两个测试用例也确实够了，如：

case4：x= 20, y= 20, 覆盖 ①③⑤⑥；

case5：x= -2, y= -2, 覆盖 ①②④⑥。

这样即可达到判定覆盖要求，具体覆盖情况如表所示。

判定覆盖

判定覆盖情况如表所示。

测试用例编号	x	y	第1个判定表达式 x>0 OR y>0	第2个判定表达式 x<10 AND y<10
case4	20	20	Y	N
case5	-2	-2	N	Y

判定覆盖

如果测试达到判定覆盖，则显然程序流程的所有分支都是会被测试到的，各个分支上的所有语句都会被测试到，所以只要满足判定覆盖，就必定会满足语句覆盖，这一点从图中可以直观的看出来。

判定覆盖

在判定覆盖中，如果一个判定表达式中有多个条件，由于我们只关注这个判断表达式的最终结果，而不是每一个条件的判定结果，所以有的条件可能只取过真值或者假值，而另外一种取值根本就没有出现过，如果这个条件写错了，那么判定覆盖测试显然是发现不了的。

条件覆盖

条件覆盖就是要求判断表达式中的每一个条件都要至少取得一次真值和一次假值，需要注意的是，每一个条件都要至少取得一次真值和一次假值并不等于每一个判定也都能至少取得一次真值和一次假值，即条件覆盖并不比判定覆盖强，两者的关注点不同。

例如，对于程序段P1，设计测试用例如下：

case6：x= 20, y = -20;

case7：x= -2, y = 20 。

这样即可达到条件覆盖要求，具体覆盖情况如表所示。

条件覆盖

条件覆盖情况如表所示。

测试用例编号	x	y	条件x>0	条件y>0	条件x<10	条件y<10
case6	20	-20	Y	N	N	Y
case7	-2	20	N	Y	Y	N

由于case6和case7对第1个IF语句，只覆盖了Y分支，对第2个IF语句，只覆盖了N分支，因此并不满足判定覆盖。

条件 / 判定覆盖

条件覆盖并不比判定覆盖强，两者只是关注点不同，有时会把条件覆盖和判定覆盖结合起来使用，这被称之为条件/判定覆盖，它的含义是指：设计足够多的测试用例，使得判定表达式中每个条件的真/假取值至少都出现一次，并且每个判定表达式自身的真/假取值也都要至少出现一次。

条件 / 判定覆盖

对于程序段P1，我们在做判定覆盖时设计的测试用例：

case4：x= 20, y= 20；

case5：x= -2, y= -2。

实际上也同时是满足条件/判定覆盖的，因为每个条件的真 / 假取值都出现了一次，并且每个判定的真/假取值结果也都出现了一次，具体覆盖情况如表所示。

条件 / 判定覆盖

条件 / 判定 覆盖情况如表所示。

测试用例编号	x	y	条件x>0	条件y>0	条件 x<10	条件 y<10
case4	20	20	Y	Y	N	N
case5	-2	-2	N	N	Y	Y

条件/判定覆盖

再来看一个三角形判定问题的案例，有程序段P2如下：

```
if ((a<b+c) && (b<a+c) && (c<a+b))
        is_Triangle = true;
else
        is_Triangle = false;
```

对该程序段进行测试时，如果要满足条件/判定覆盖，则四个条件表达式都要既有true取值，也有false取值。四个条件表达式如表所示。

条件 / 判定覆盖

四个条件表达式如表所示。

条件表达式编号	条件表达式
1	a<b+c
2	b<a+c
3	c<a+b
4	(a<b+c) && (b<a+c) && (c<a+b)

条件 / 判定覆盖

设计测试用例如下：

case8 ： a=1，b=1，c=1；
case9 ： a=1，b=2，c=3；
case10: a=3，b=1，c=2；
case11: a=2，b=3，c=1。

即可满足条件/判定覆盖，具体覆盖情况如表所示。

条件 / 判定覆盖

条件/判定覆盖情况如表所示。

测试用例编号	a	b	c	条件表达式1	条件表达式2	条件表达式3	条件表达式4
Case8	1	1	1	Y	Y	Y	Y
Case9	1	2	3	Y	Y	N	N
Case10	3	1	2	N	Y	Y	N
Case11	2	3	1	Y	N	Y	N

条件组合覆盖

条件组合覆盖，它的含义是：设计足够多的测试用例，使得每个判定中条件取值的各种组合都至少出现一次。

显然满足条件组合覆盖的测试用例一定也是满足判定覆盖、条件覆盖和条件/判定组合覆盖的。

条件组合覆盖

对于程序段P1，由于一个判定中有两个条件，而两个条件可能的组合情况有4种，所以，如果要达到条件组合覆盖，至少需要四个测试用例。如果能够合理设计，让四个测试用例在覆盖第1个判定四种条件组合的同时也覆盖第2个判定的四种条件组合，那么四个测试用例就够了。设计的测试用例如下：

case12：x= 50, y= 50；

case13：x= -5, y= -5；

case14：x= 50, y= -5；

case15：x= -5, y= 50；

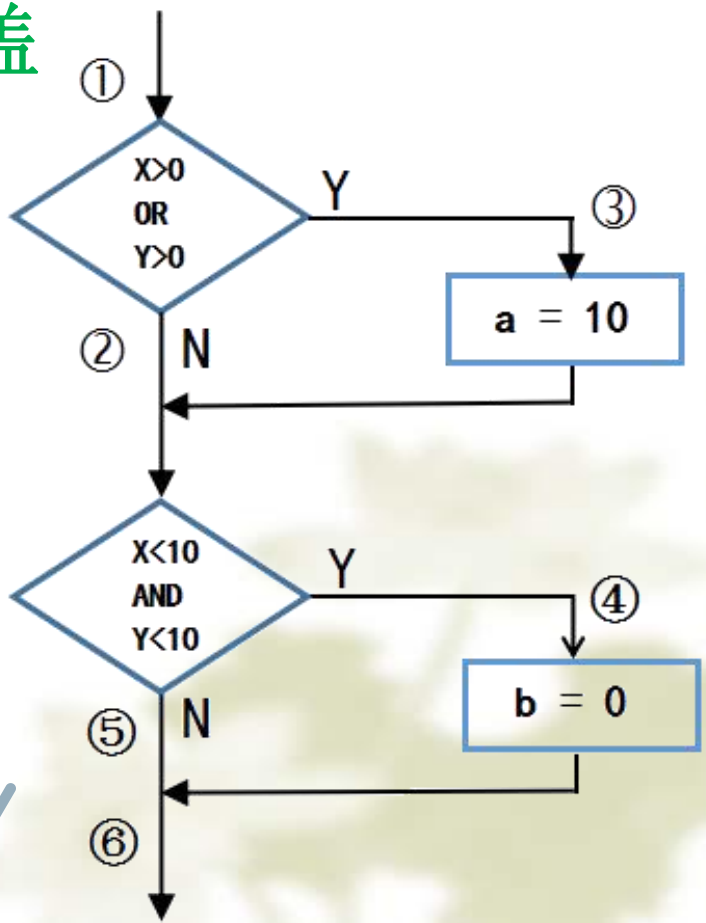
条件组合覆盖

对条件组合的覆盖情况如表所示。

测试用例编号	x	y	第1个判定		第2个判定	
			条件 x>0	条件 y>0	条件 x<10	条件 y<10
case12	50	50	Y	Y	N	N
case13	-5	-5	N	N	Y	Y
case14	50	-5	Y	N	N	Y
case15	-5	50	N	Y	Y	N

条件组合覆盖

上面四个例子虽然满足条件组合覆盖，但并不能覆盖程序中的每一条路径，例如路径①②⑤⑥就没有执行过，因此，条件组合覆盖标准仍然可能是不彻底的。



本节内容就讲到这里，谢谢，再见！

金陵科技學院